Capstone Project

Certification Project – Finance Me

Banking and Finance Domain

Submitted by: Sumit Kumar Singh

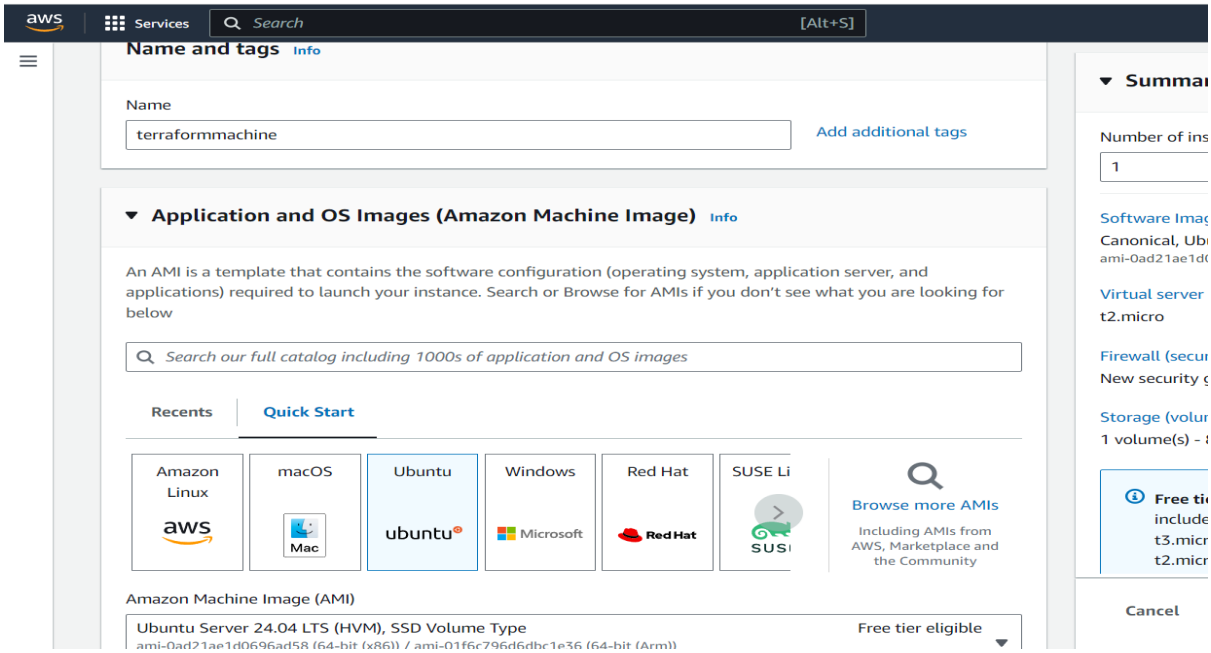
Date Of Submission: 08/08/2024

Submitted to: Mr. Vikul

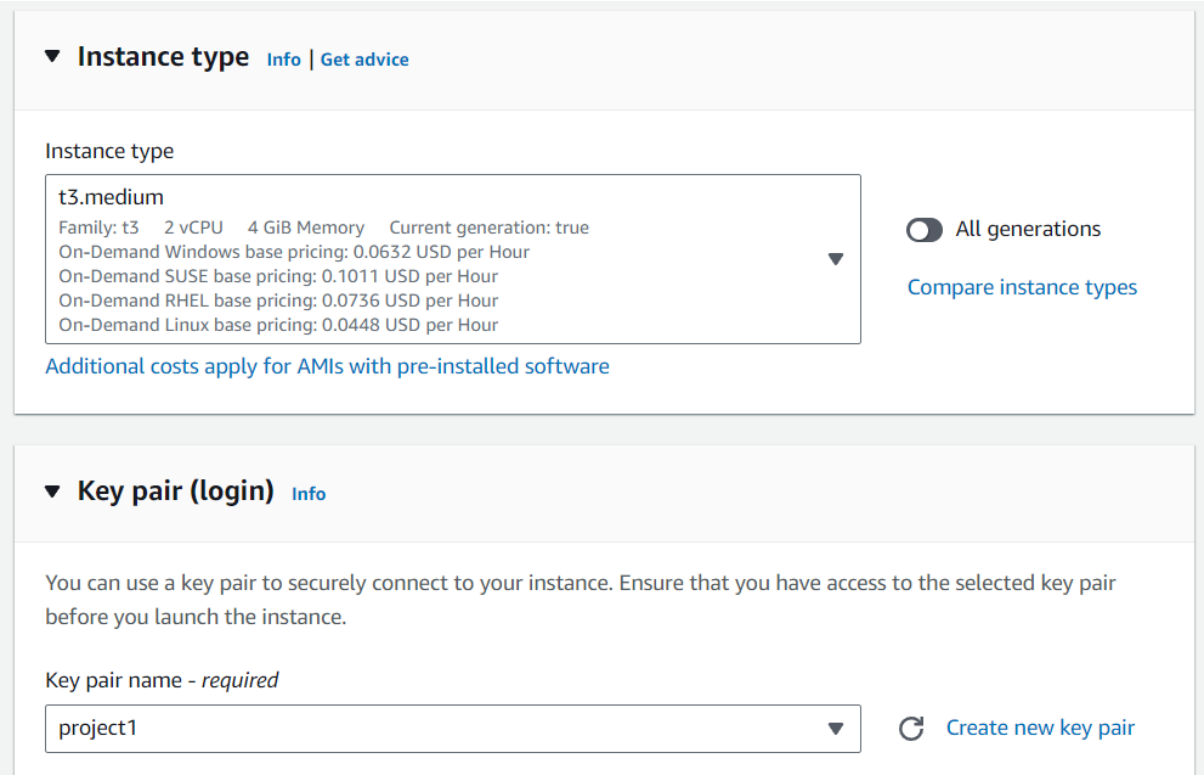
Banking and finance domain project:

Step1: Go to the AWS console, Launched one instance,in this instance we configure terraform

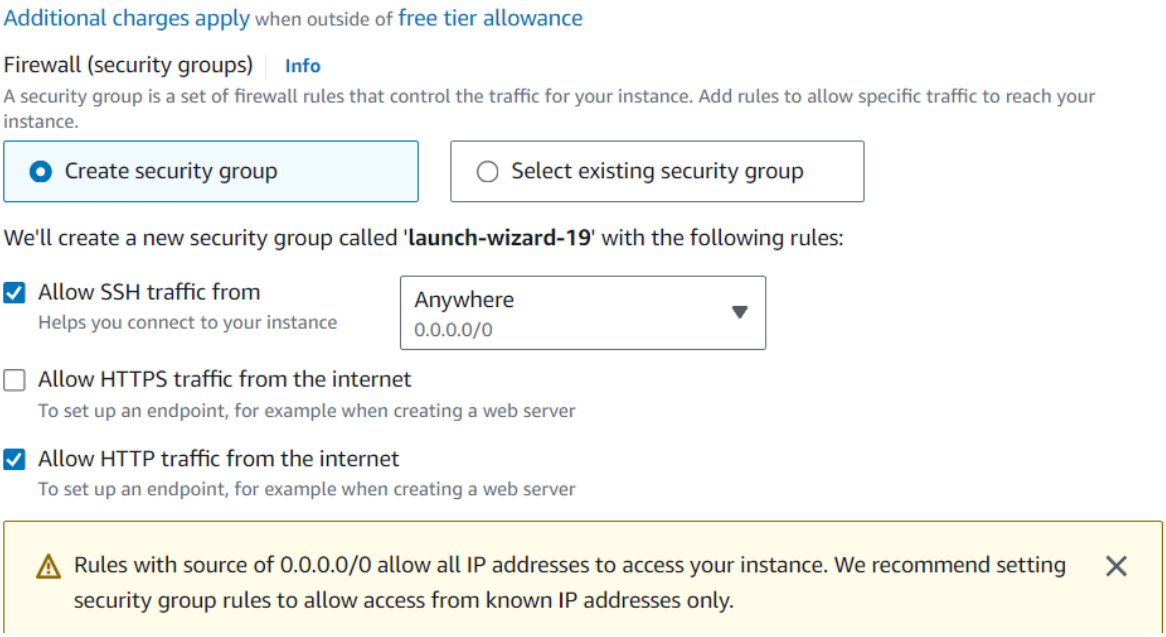
- **Select ubuntu**
- **In AMI SELECT latest version**



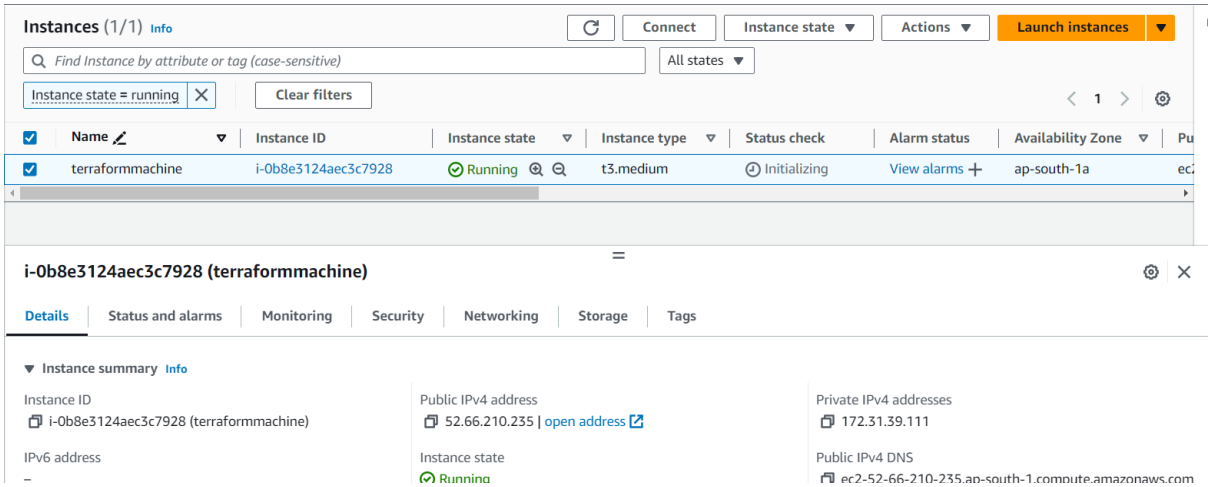
- **In instance type select t3.medium**



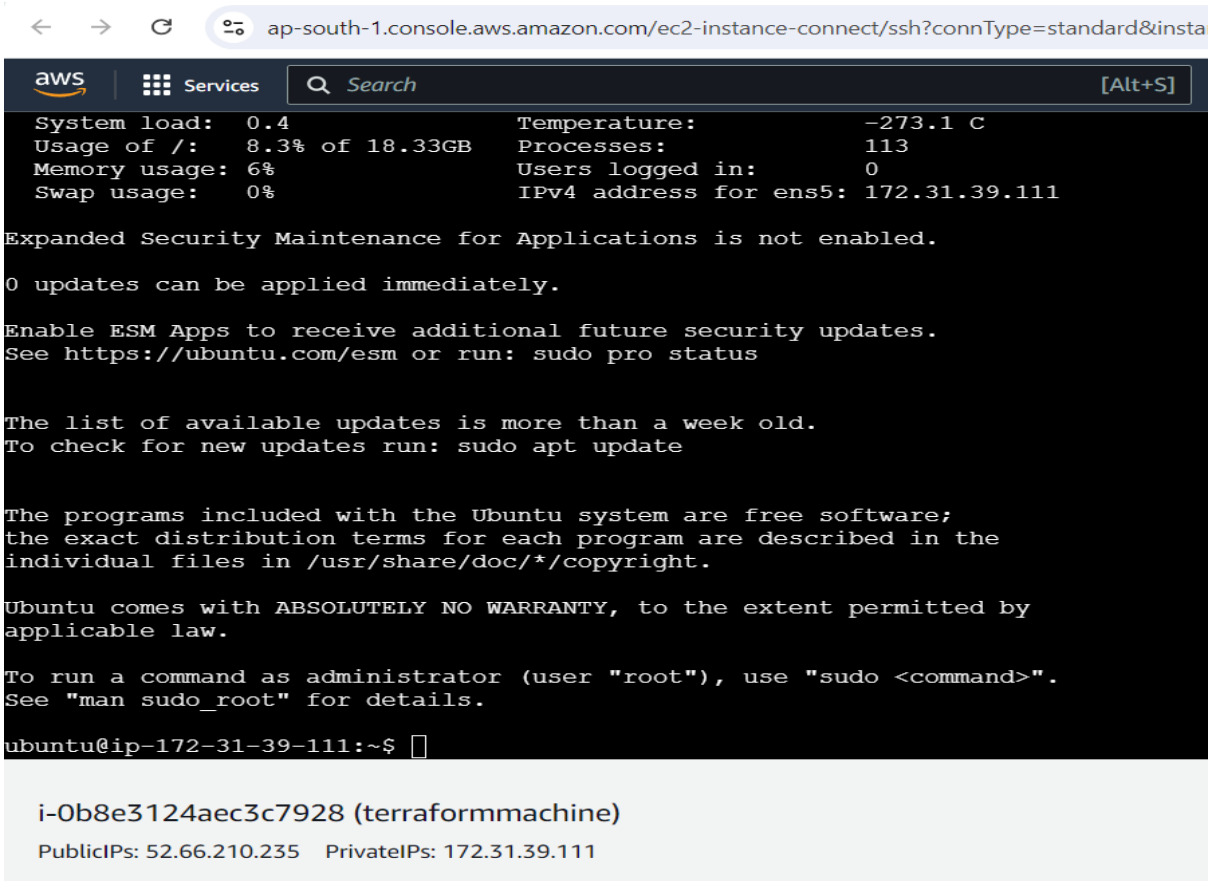
- **In network setting,click on edit, Allow all traffic from anywhere**



- **As you can see, an Instance has been created**



- **Connect the instance**

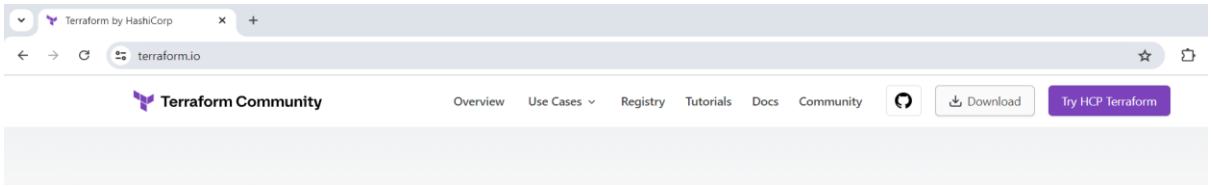


Step 2: Execute below commands in instances

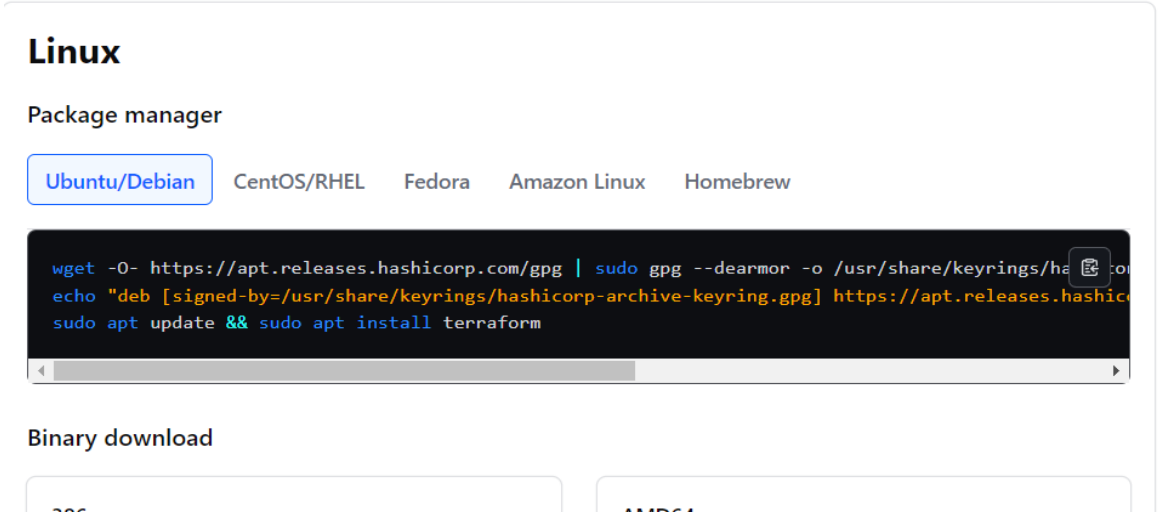
$sudo su

Step 3: Now Search terraform.io in browser

- **Click on download**



- **Scroll down come to linux package manager,there is some code need to execute one by one for install terraform**

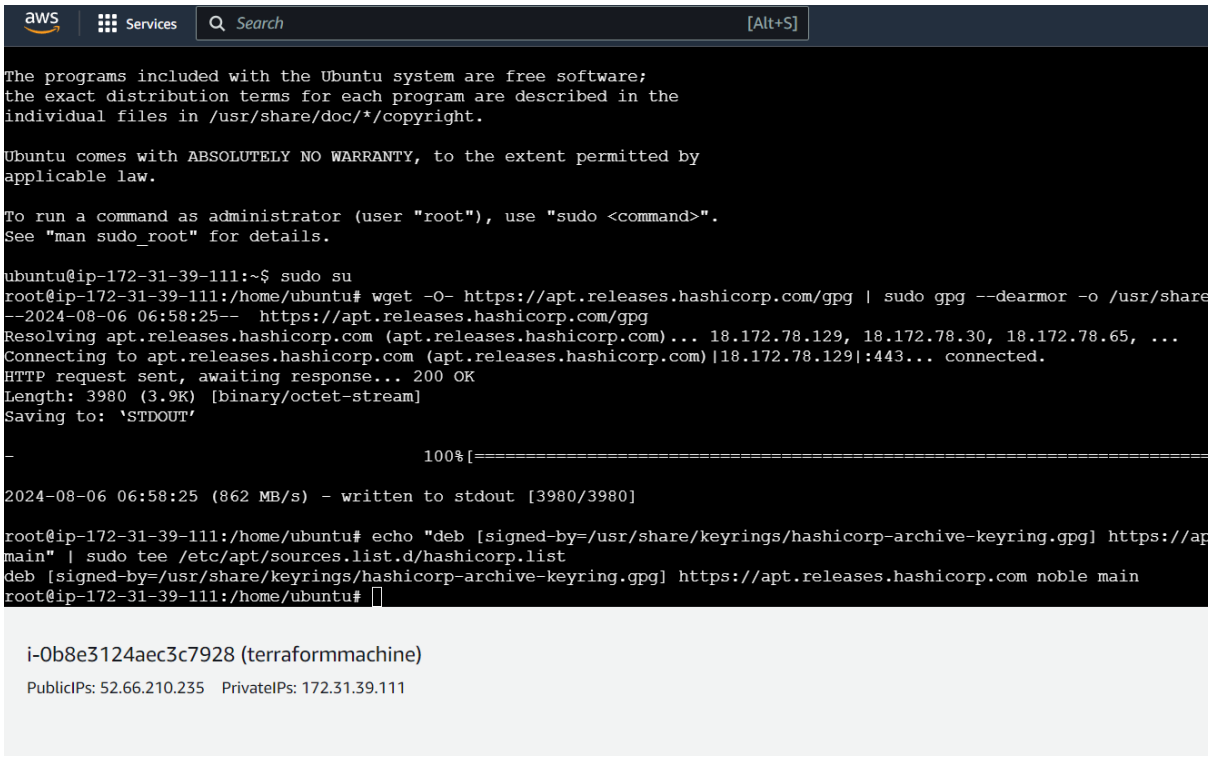


- **Paste first below command**

$wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg

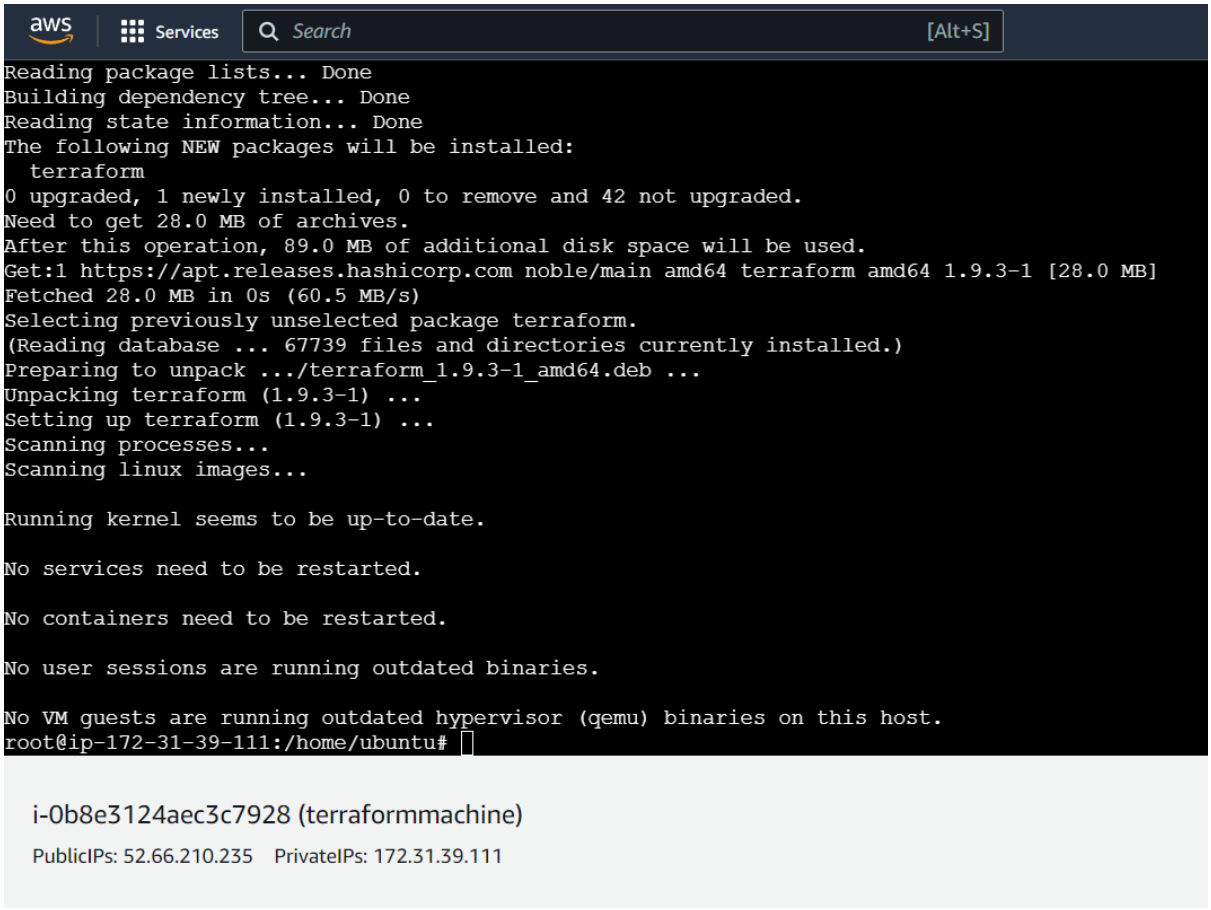
- **Paste Second below command**

$echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list



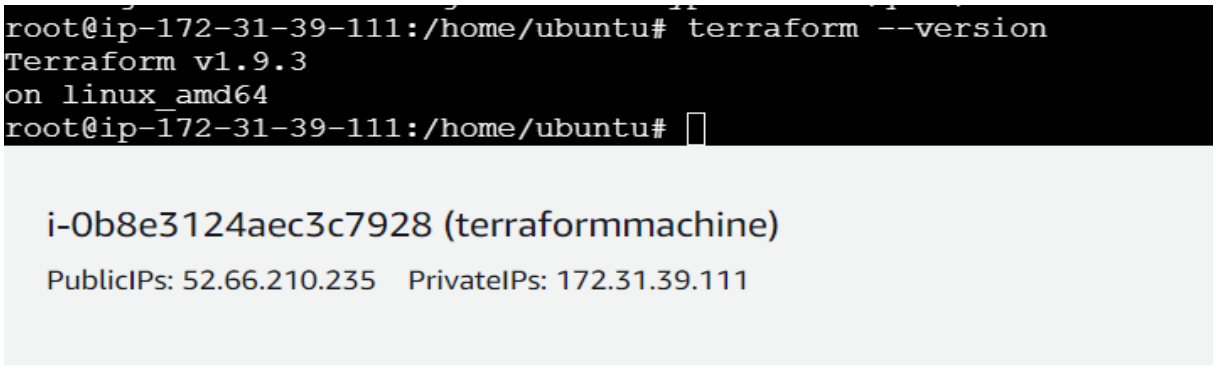
- **Now Paste Third below command**

$sudo apt update && sudo apt install terraform



Step 4: Now execute below command For check terraform version

$terraform - -version



- **You can see we have installed terraform**

- **Step 5: Now we will Create a directory by execute below command**

$mkdir project

$cd project

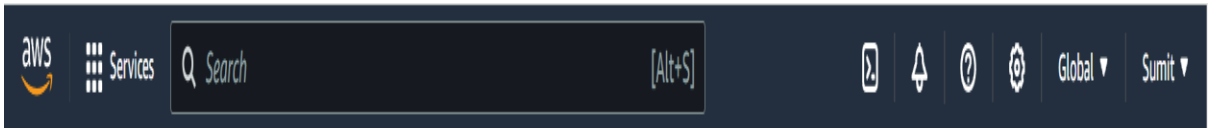
```
root@ip-172-31-39-111:/home/ubuntu# mkdir project
root@ip-172-31-39-111:/home/ubuntu# ls
project
root@ip-172-31-39-111:/home/ubuntu# cd project/
root@ip-172-31-39-111:/home/ubuntu/project#
```

i-0b8e3124aec3c7928 (terraformmachine)

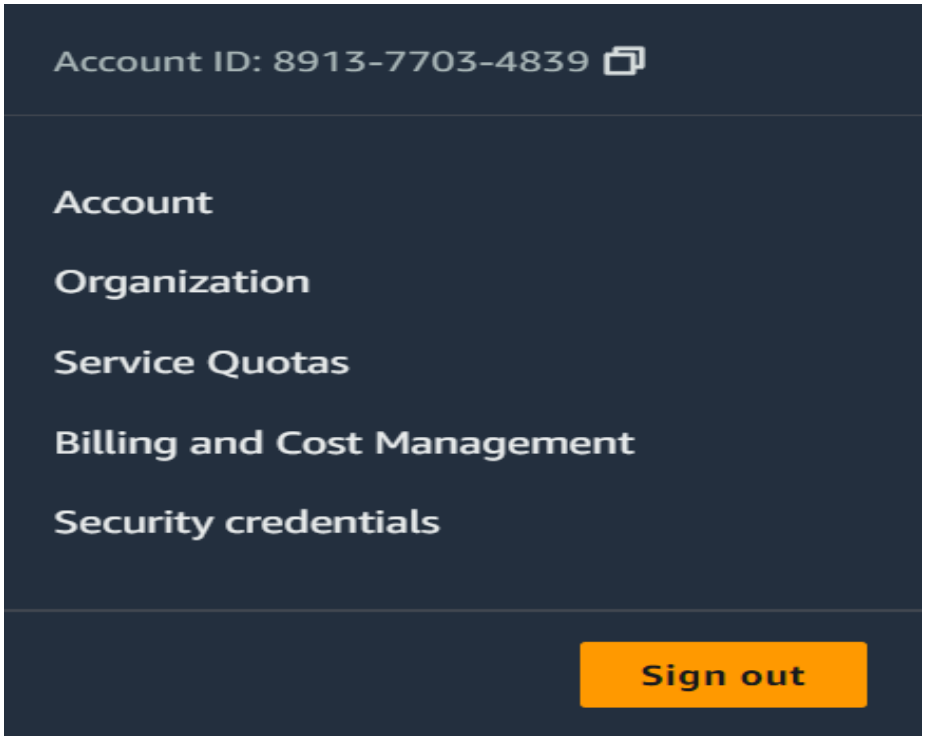
PublicIPs: 52.66.210.235 PrivateIPs: 172.31.39.111

Step 6: Now we will generate the access key and secret access key

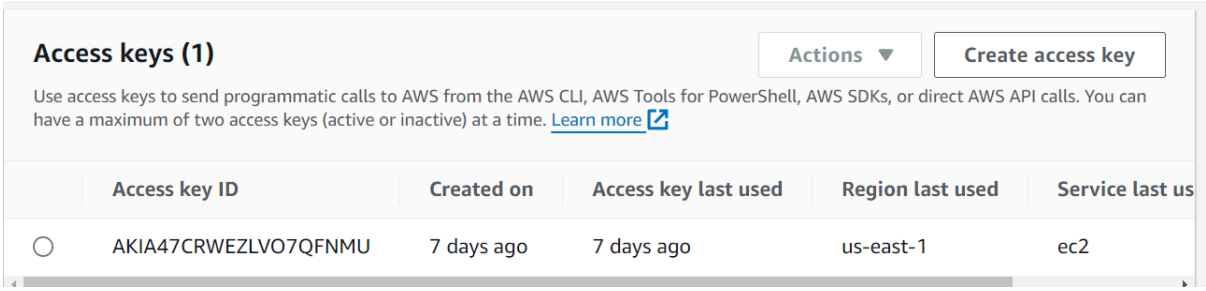
- **Go to aws console page,click on upper right corner**



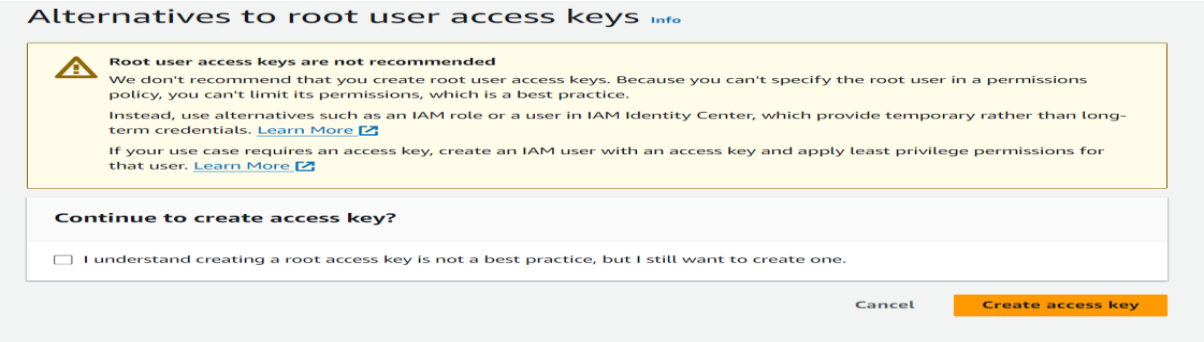
- **Now click on security credentials**



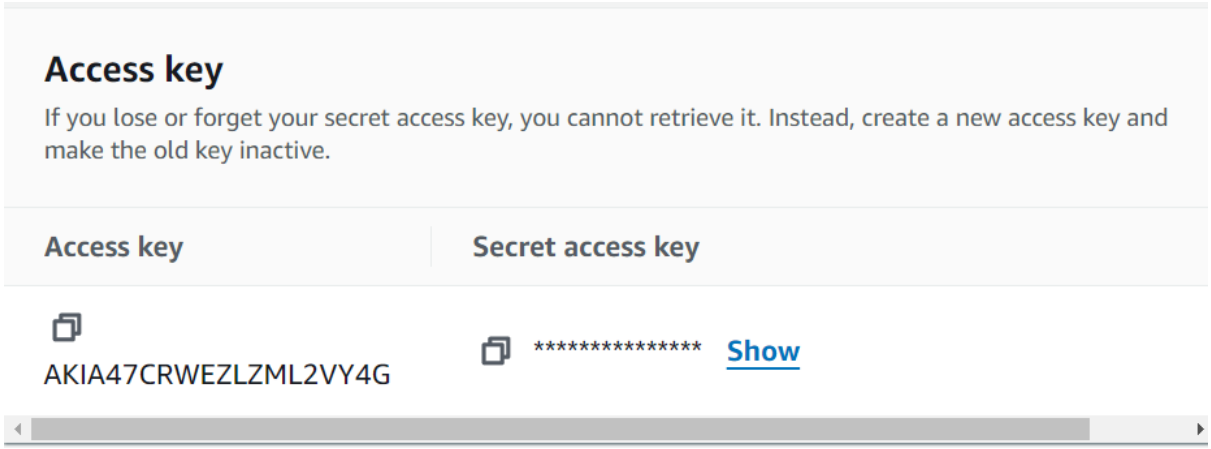
- **Scroll down there is a option of Access keys,click on create access key**



- **Now tick on check box and again click on create access key**

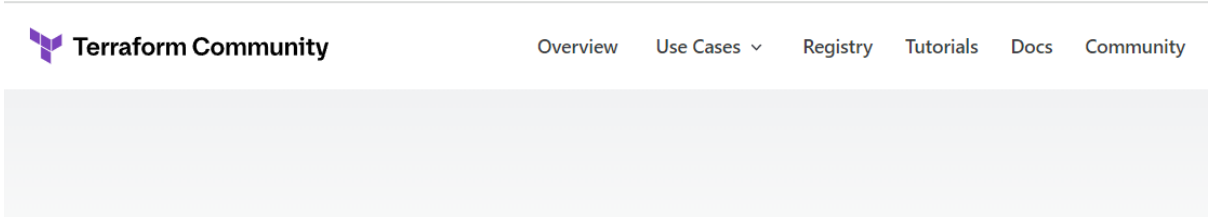


- **You can see I have successfully generated the access key**



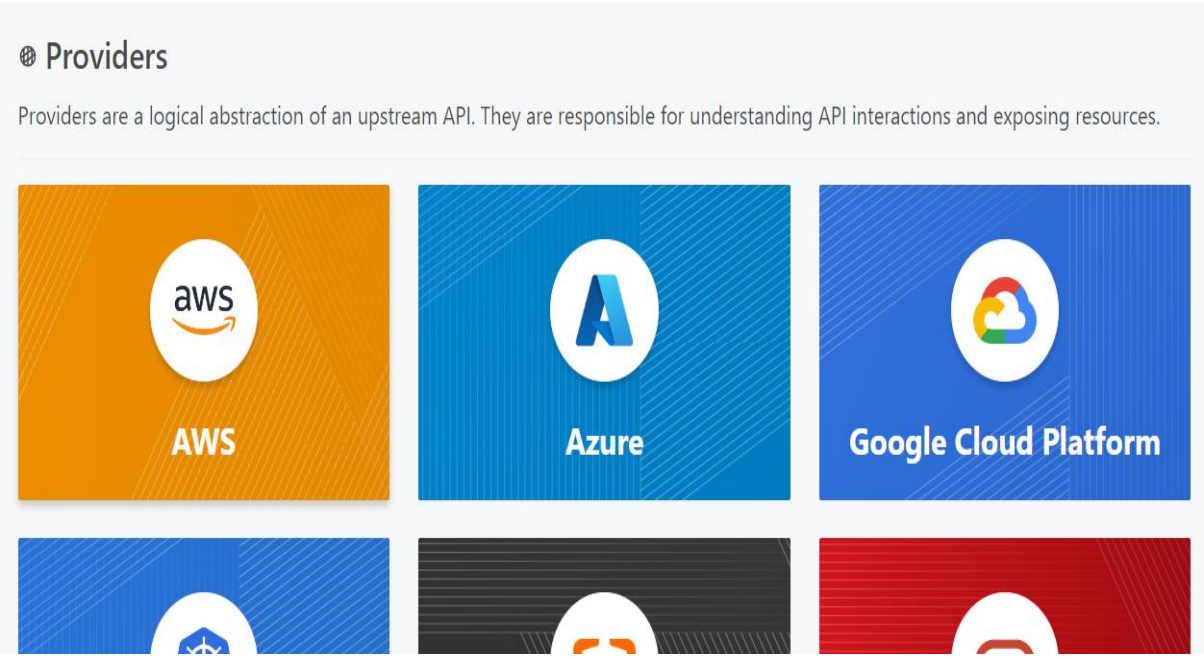
Step 7: Now again go to terraform.io page on the browser

- **Click on download registry**

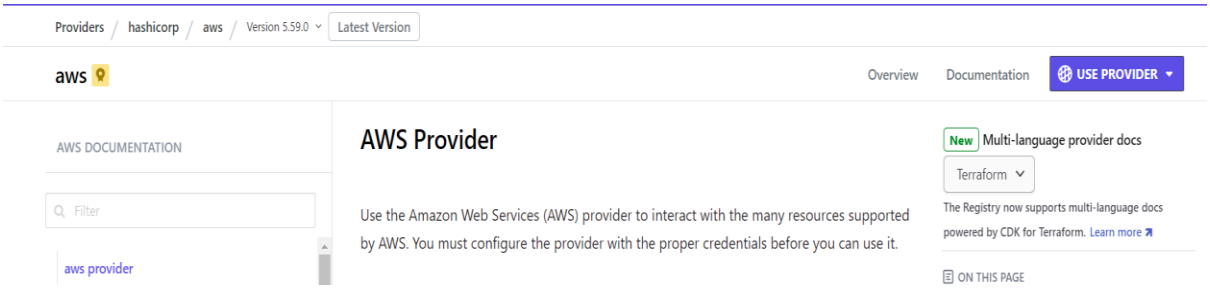


- **Click on browse provider**

- **Click on AWS**



- **Click on Documentation**



- **Scroll down and go to the usage section ,copy the code for configure access key and secret access key)**

Usage:

```
provider "aws" {
  region     = "us-west-2"
  access_key = "my-access-key"
  secret_key = "my-secret-key"
}
```

Copy

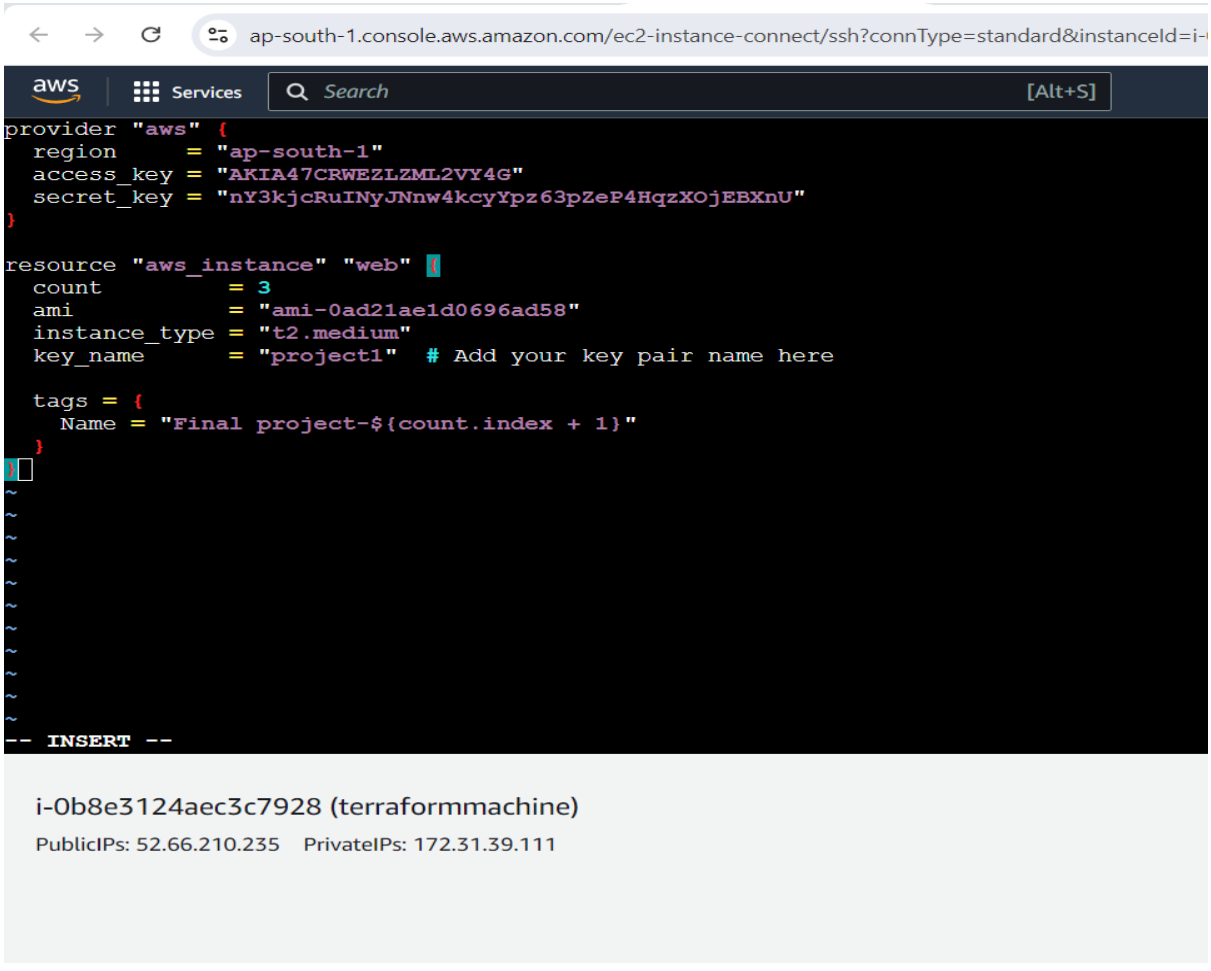
Other settings related to authorization can be configured, such as:

- `profile`

Step 8: Create a file with name ec2.tf in instance by execute below command(terraform file is always ends with .tf)

$vi ec2.tf

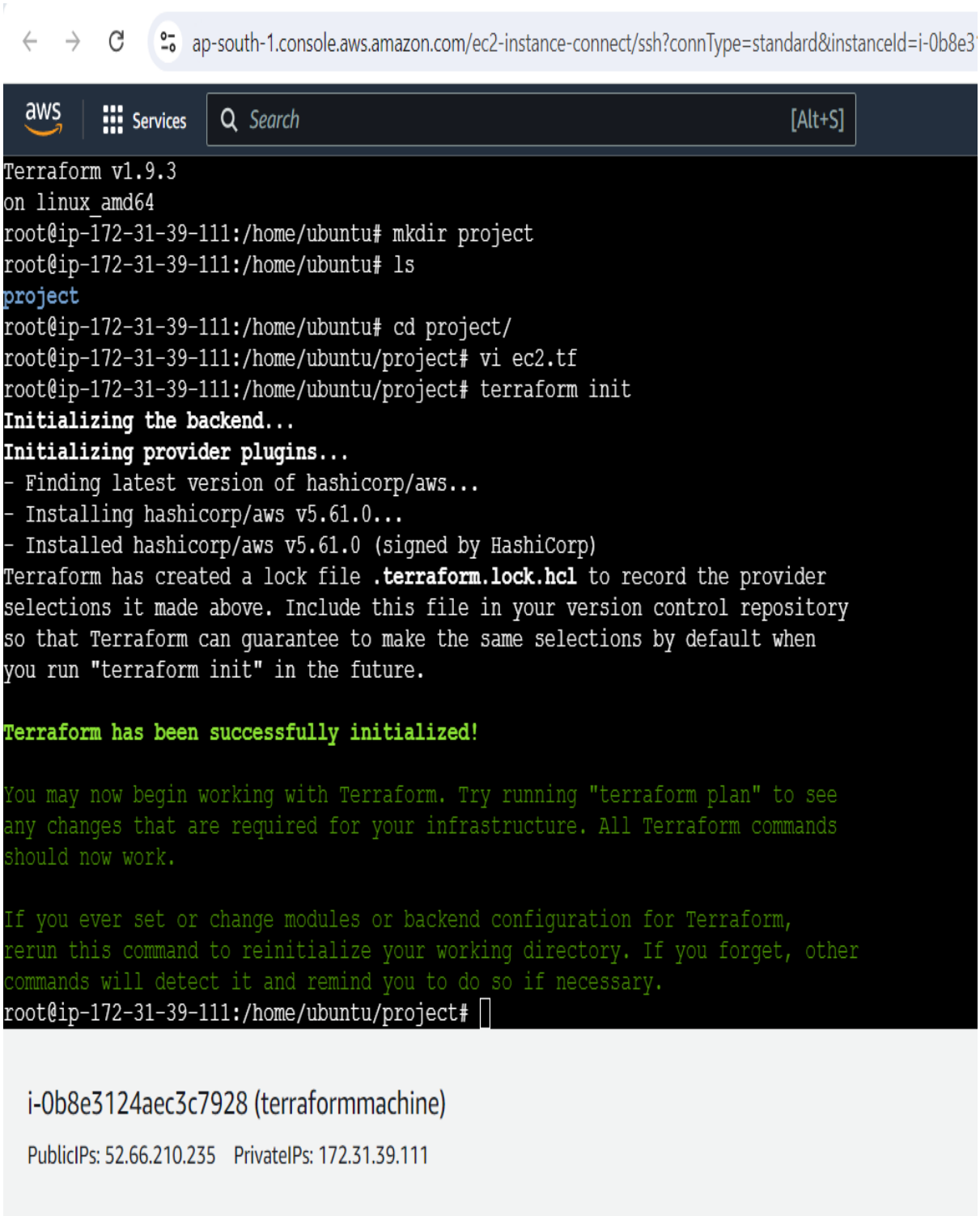
- In this we use provider as aws
- I have pasted my access key and secret access key
- In resource we write aws instance
- Count as 3(because we are creating three instances)
- Pasted ubuntu ami id
- In instance type we write t3.medium
- And in tag we write the instance name



- **Press esc--->:wq**

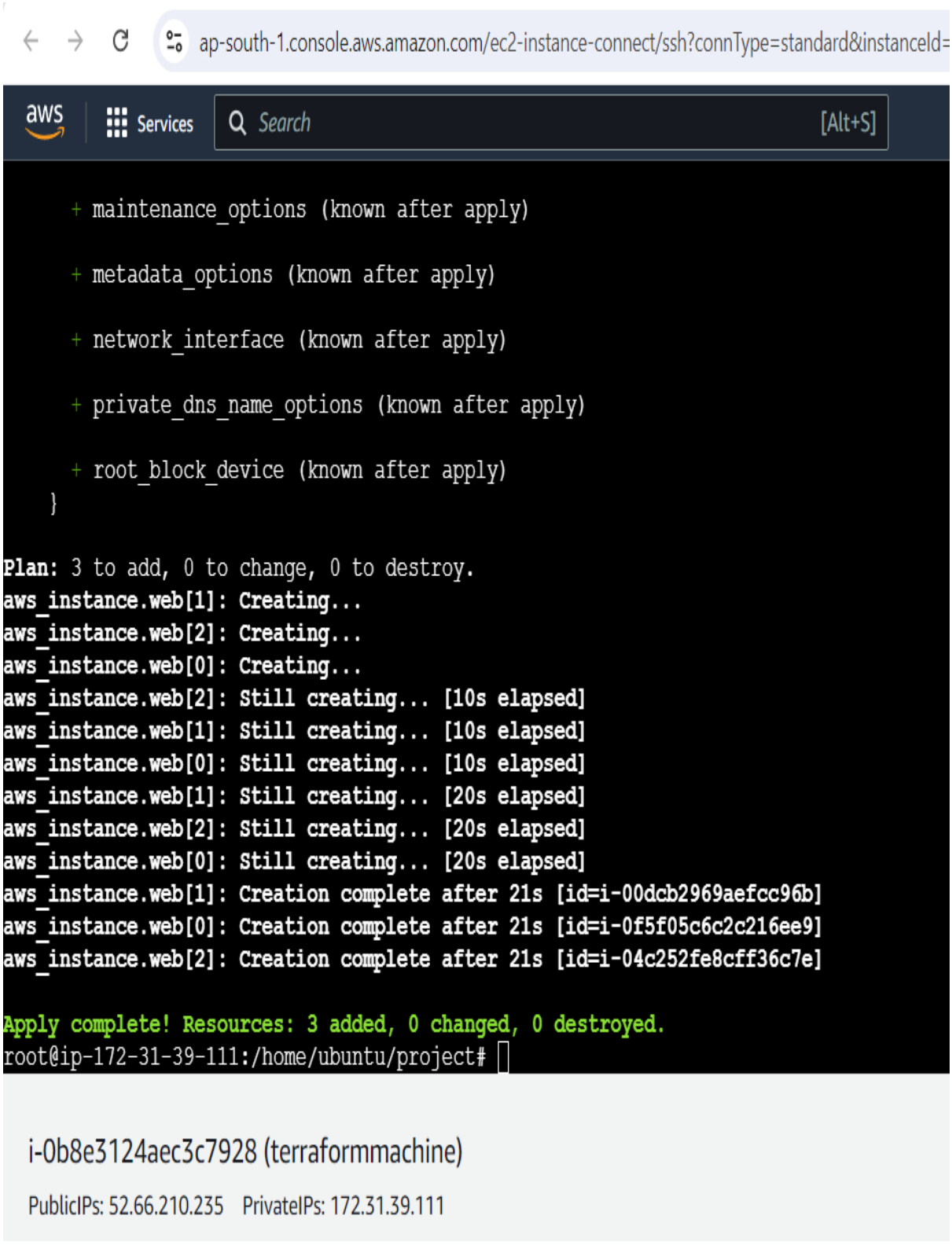
Step 9: Now execute below command for initialize the terraform

$terraform init



Step 10: Now execute below command to execute ec2.tf file

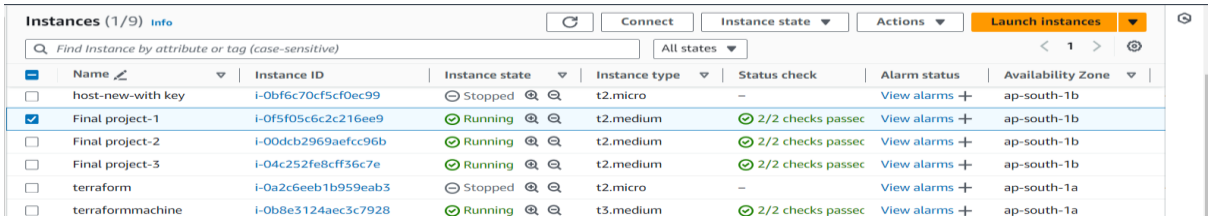
$terraform apply - -auto-approve



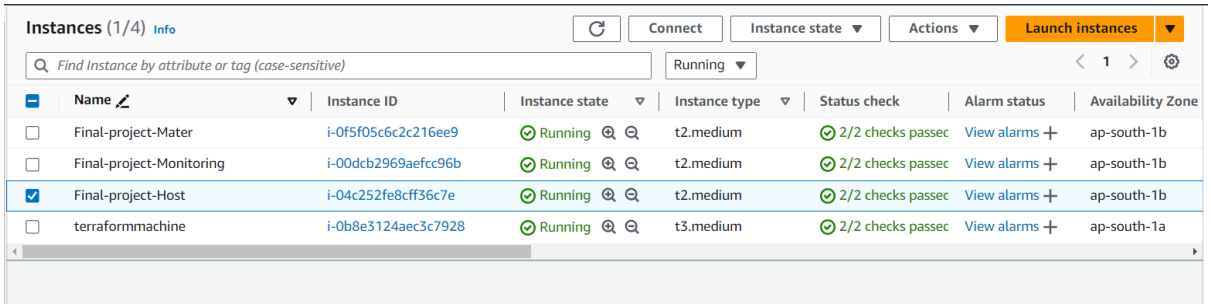
- **You can see we have added 3 resources**

Step 11: For verify instance has been created go to ec2 dashboard,there you can see we have created 3 instances via terraform

- Rename the instance as Final-project-Master(In which we install Jenkins)
- Final-project-Host
- Final-project-Monitoring(Prometheus and Grafana we install in this)

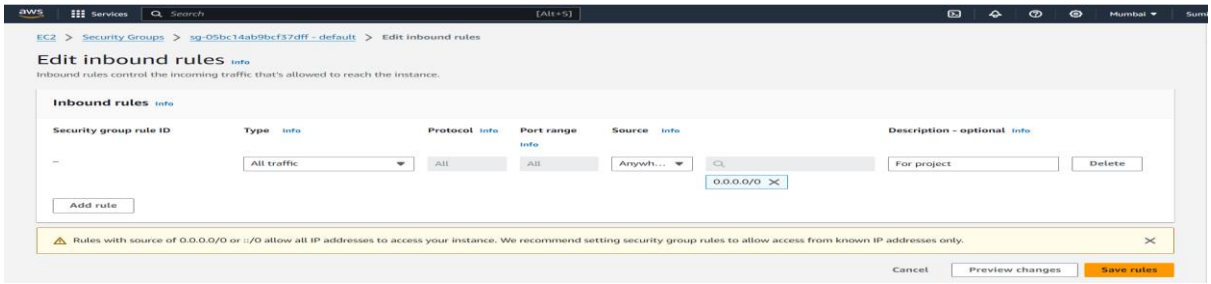


- Rename created instance

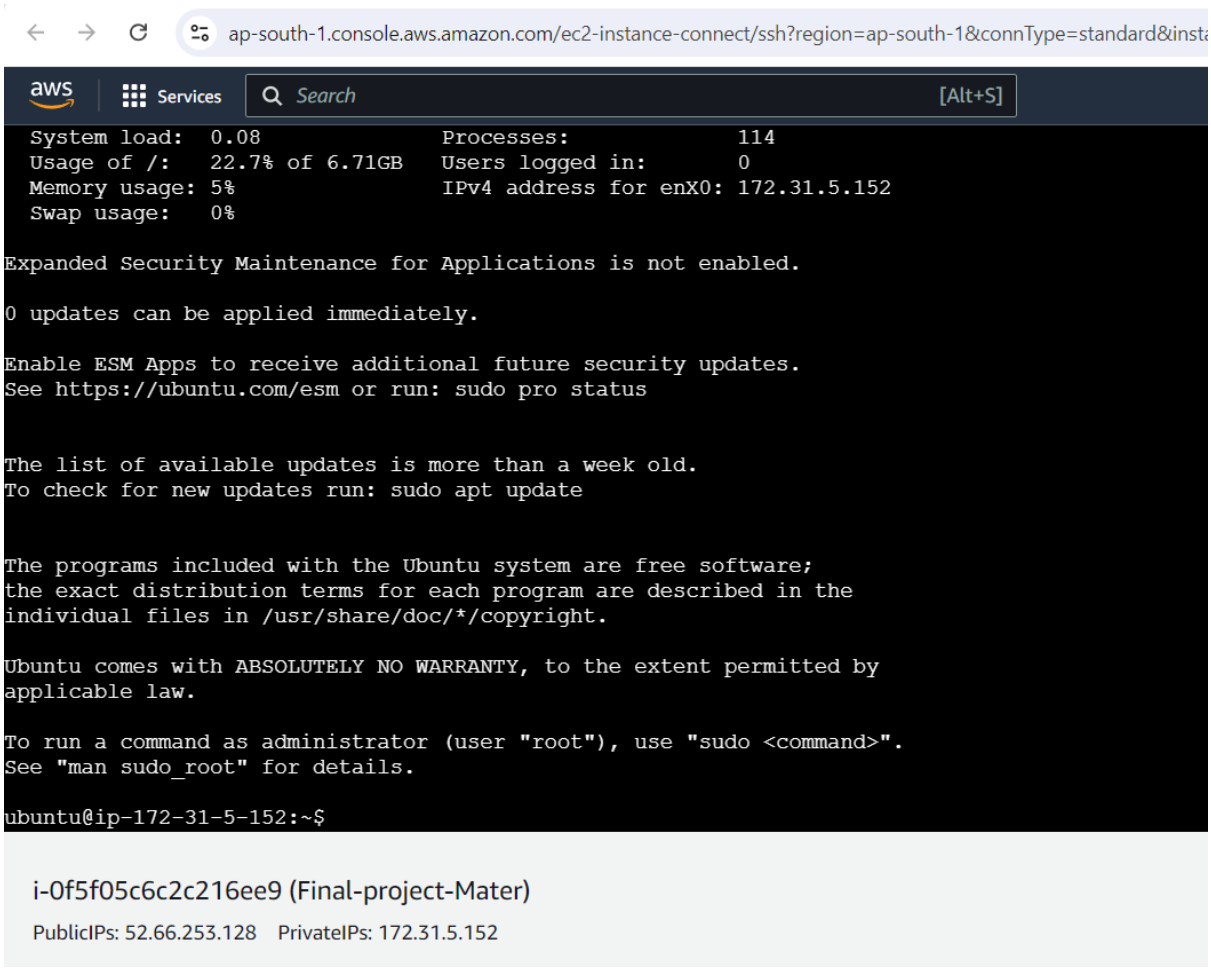


Step 12: Now edited the security group of all three instances as below

- Go to inbound rules delete existing rule and add new one

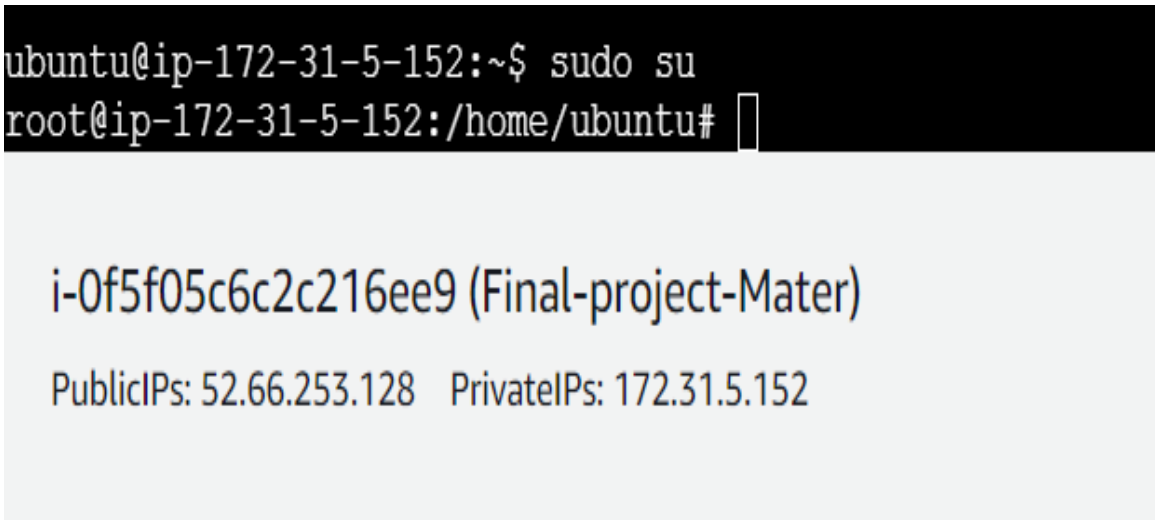


Step 13: Now connect the final-project-master instance



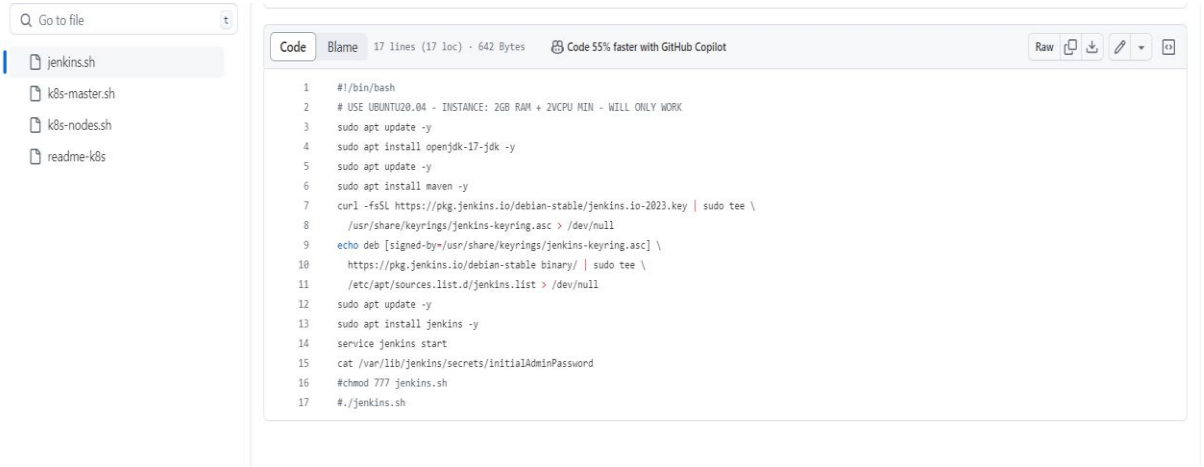
- **Now execute below command for become root user**

$sudo su

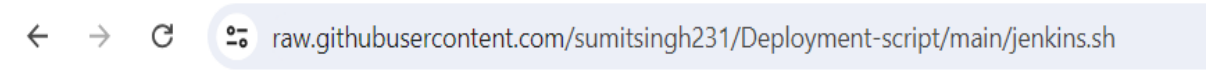


Step 14: Now we will install Jenkins in this machine ,for install Jenkins followed below steps

- Go to deployment script github repository
- Click on Jenkins.sh file(it is a shell script file to install Jenkins)



- Click on the raw button and copy the link




```
#!/bin/bash
# USE UBUNTU20.04 - INSTANCE: 2GB RAM + 2VCPU MIN - WILL ONLY WORK
sudo apt update -y
sudo apt install openjdk-11-jdk -y
sudo apt update -y
sudo apt install maven -y
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee \
  /usr/share/keyrings/jenkins-keyring.asc > /dev/null
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt update -y
sudo apt install jenkins -y
service jenkins start
cat /var/lib/jenkins/secrets/initialAdminPassword
#chmod 777 jenkins.sh
#./jenkins.sh
```

Step 15: Now execute below command in master machine

$wget <paste the copied link>

$wget https://raw.githubusercontent.com/sumitsingh231/Deployment-script/main/jenkins.sh

```
ubuntu@ip-172-31-5-152:~$ sudo su
root@ip-172-31-5-152:/home/ubuntu# wget https://raw.githubusercontent.com/sumitsingh231/Deployment-script/main/jenkins.sh
--2024-08-06 07:31:40--  https://raw.githubusercontent.com/sumitsingh231/Deployment-script/main/jenkins.sh
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.111.133, 185.199.108.133, 185.199.109.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 642 [text/plain]
Saving to: 'jenkins.sh'

jenkins.sh                          100%[=============================================================================>]     642  --.-KB/s    in 0s

2024-08-06 07:31:40 (27.6 MB/s) - 'jenkins.sh' saved [642/642]

root@ip-172-31-5-152:/home/ubuntu#
```

i-0f5f05c6c2c216ee9 (Final-project-Mater)

PublicIPs: 52.66.253.128 PrivateIPs: 172.31.5.152

Step 16: Now change the mode for giving read write execute permission for this execute below command

$chmod +x jenkins.sh

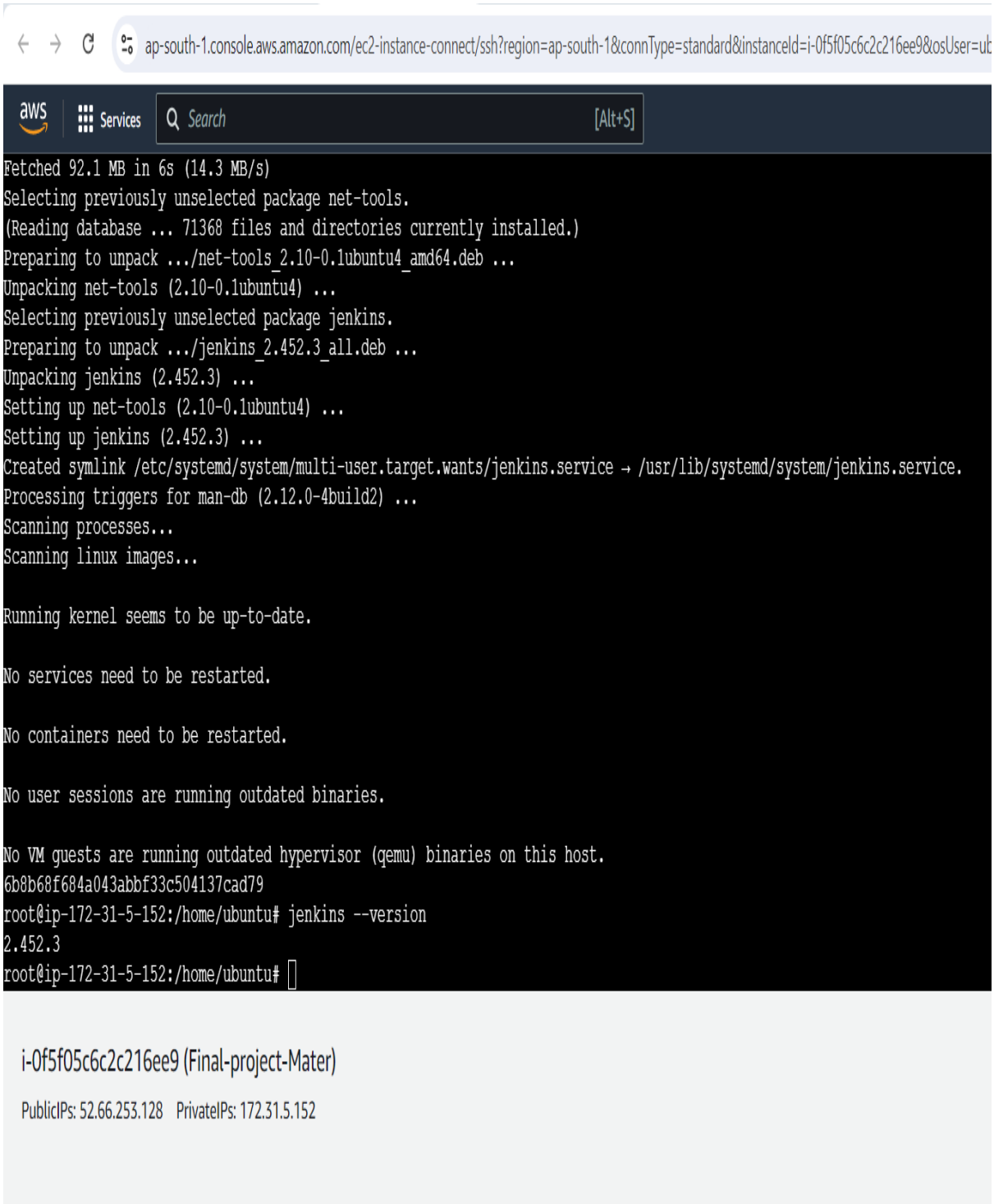
```
root@ip-172-31-5-152:/home/ubuntu# chmod +x jenkins.sh
root@ip-172-31-5-152:/home/ubuntu# ls
jenkins.sh
root@ip-172-31-5-152:/home/ubuntu# ls -l
total 4
-rwxr-xr-x 1 root root 642 Aug  6 07:31 jenkins.sh
root@ip-172-31-5-152:/home/ubuntu#
```

i-0f5f05c6c2c216ee9 (Final-project-Mater)

PublicIPs: 52.66.253.128 PrivateIPs: 172.31.5.152

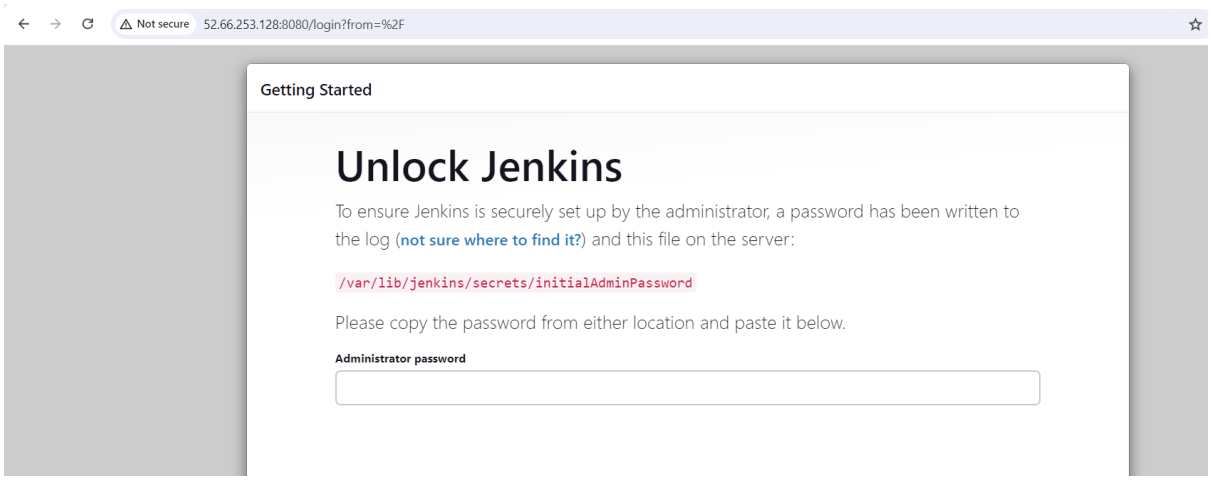
Step 17: Now execute below command to run the Jenkins,sh shell script

$./jenkins.sh



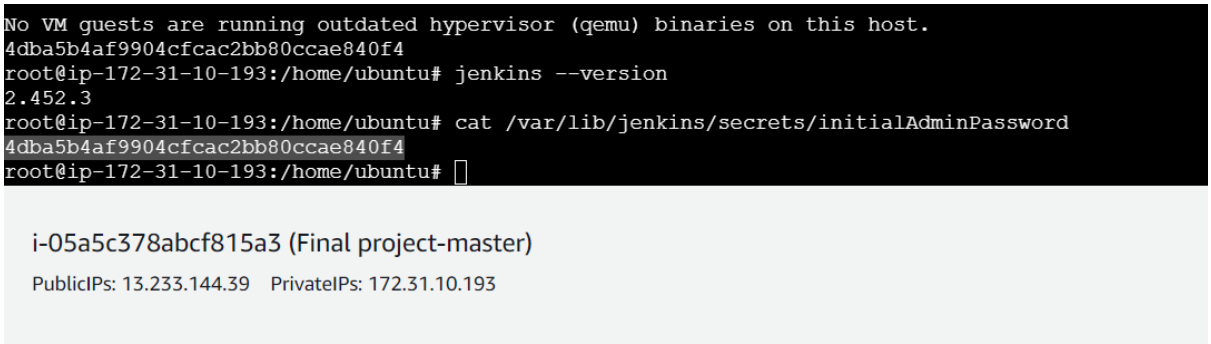
- **You can see we have installed Jenkins**

Step 18: Now copy the public ip of Jenkins master machine and paste it into the browser with:8080(Jenkins default port)

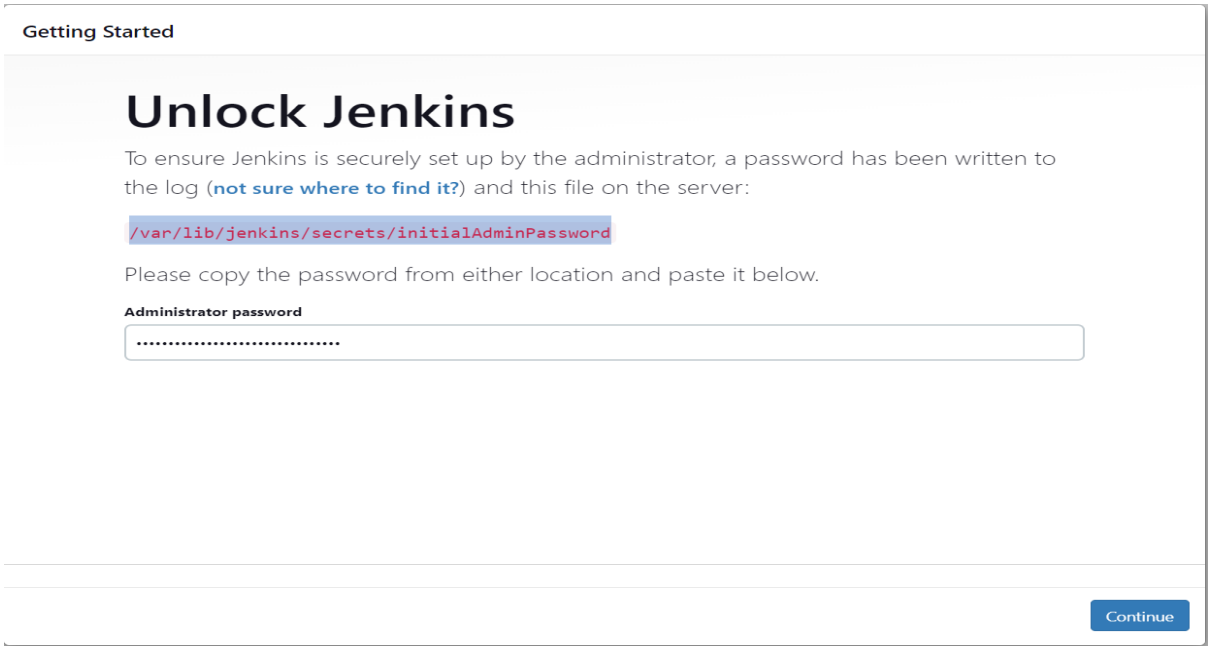


Step 19: copy the highlighted path and execute below command to see administrator password

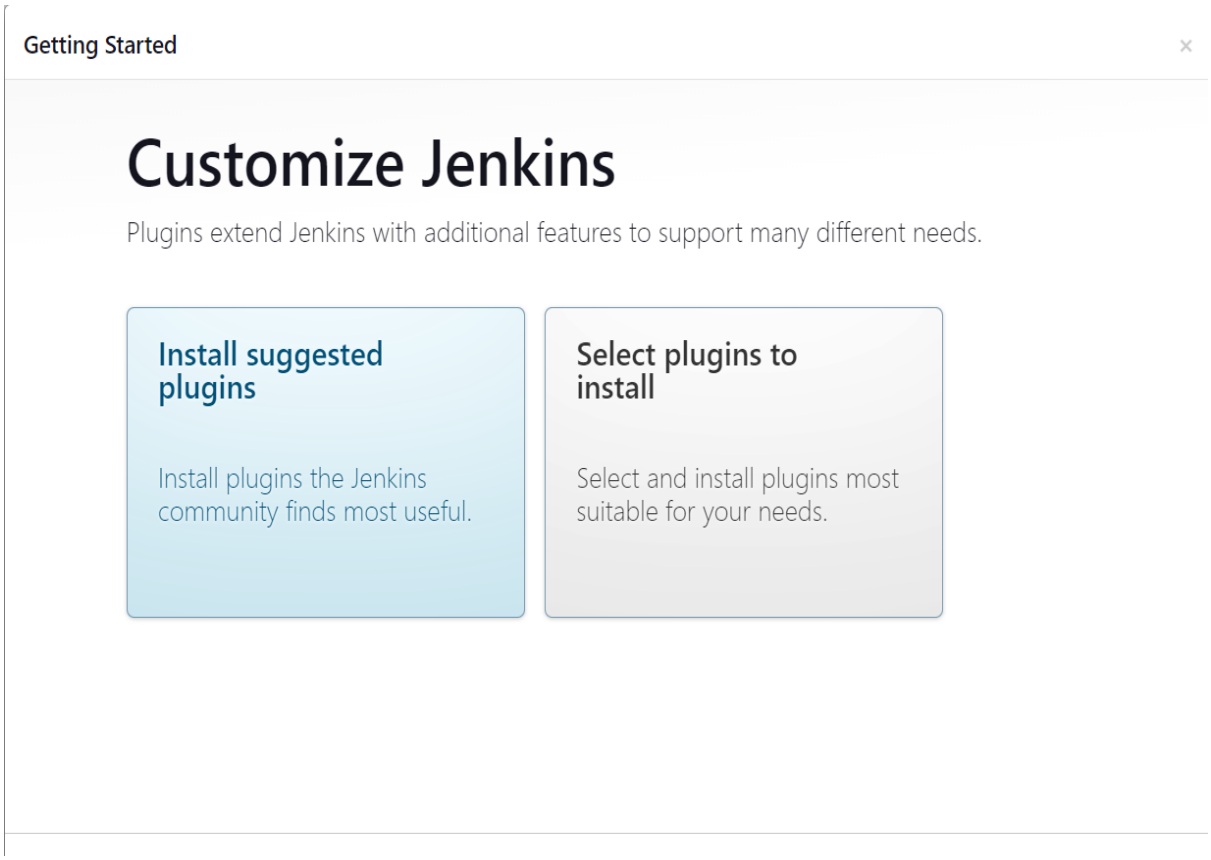
$cat /var/lib/jenkins/secrets/initialAdminPassword



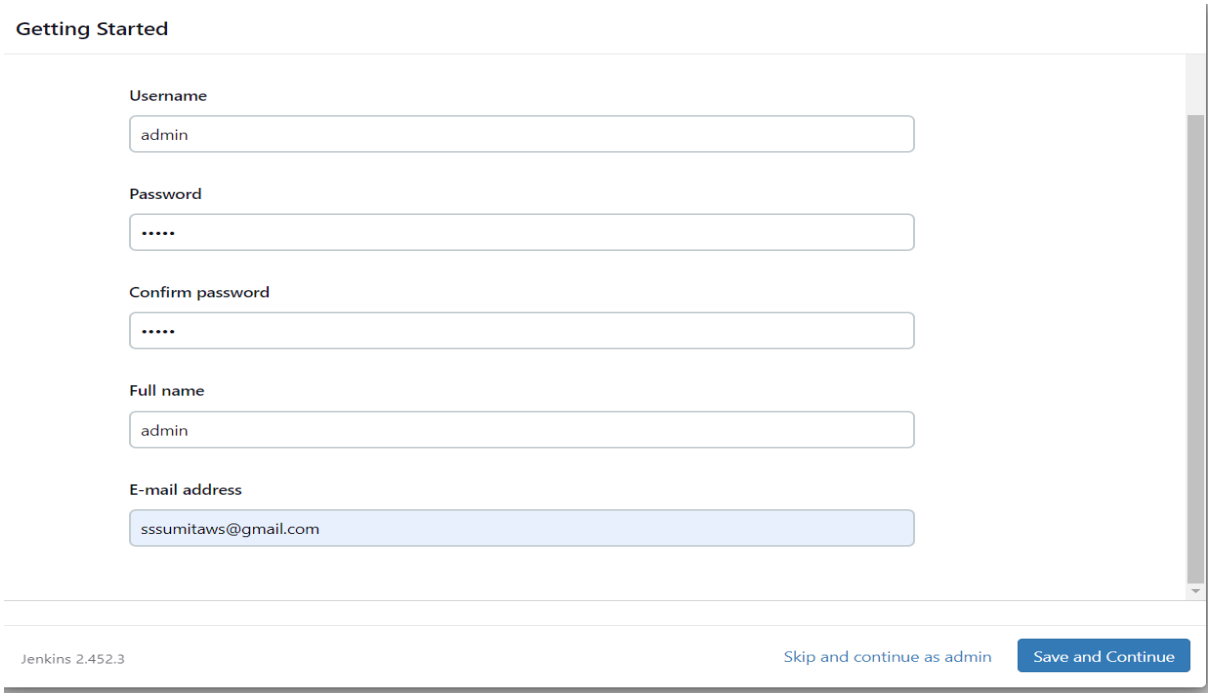
- **Paste the password into administrator password section then click on continue**



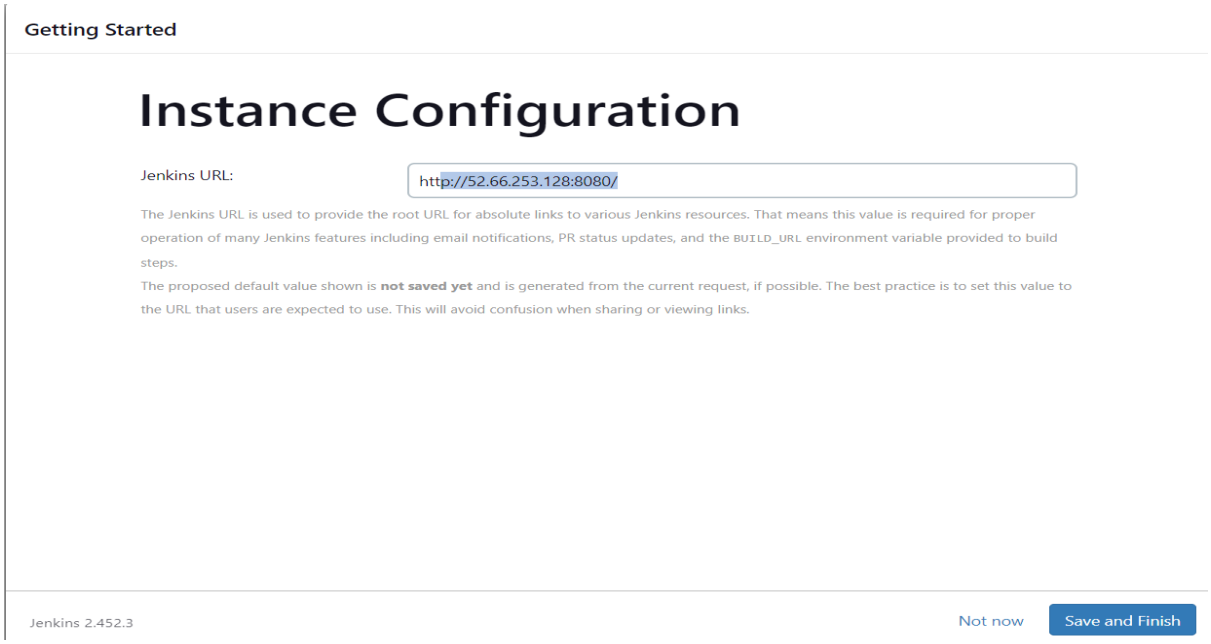
- **Click on install suggested plugins**



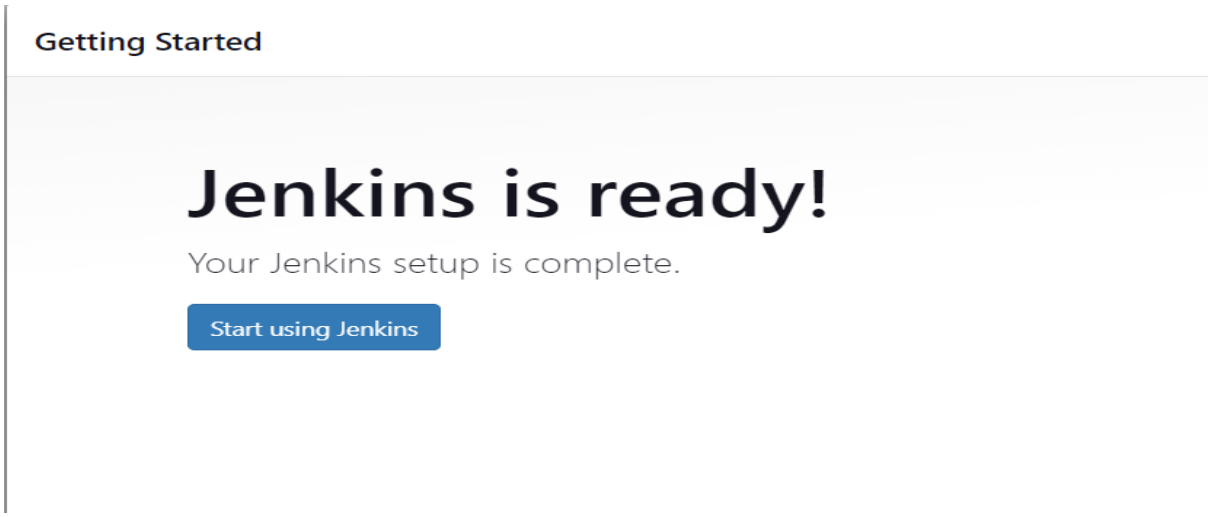
- **You can see plugins has been installed and Jenkins getting started,in this page used user name and password as admin**



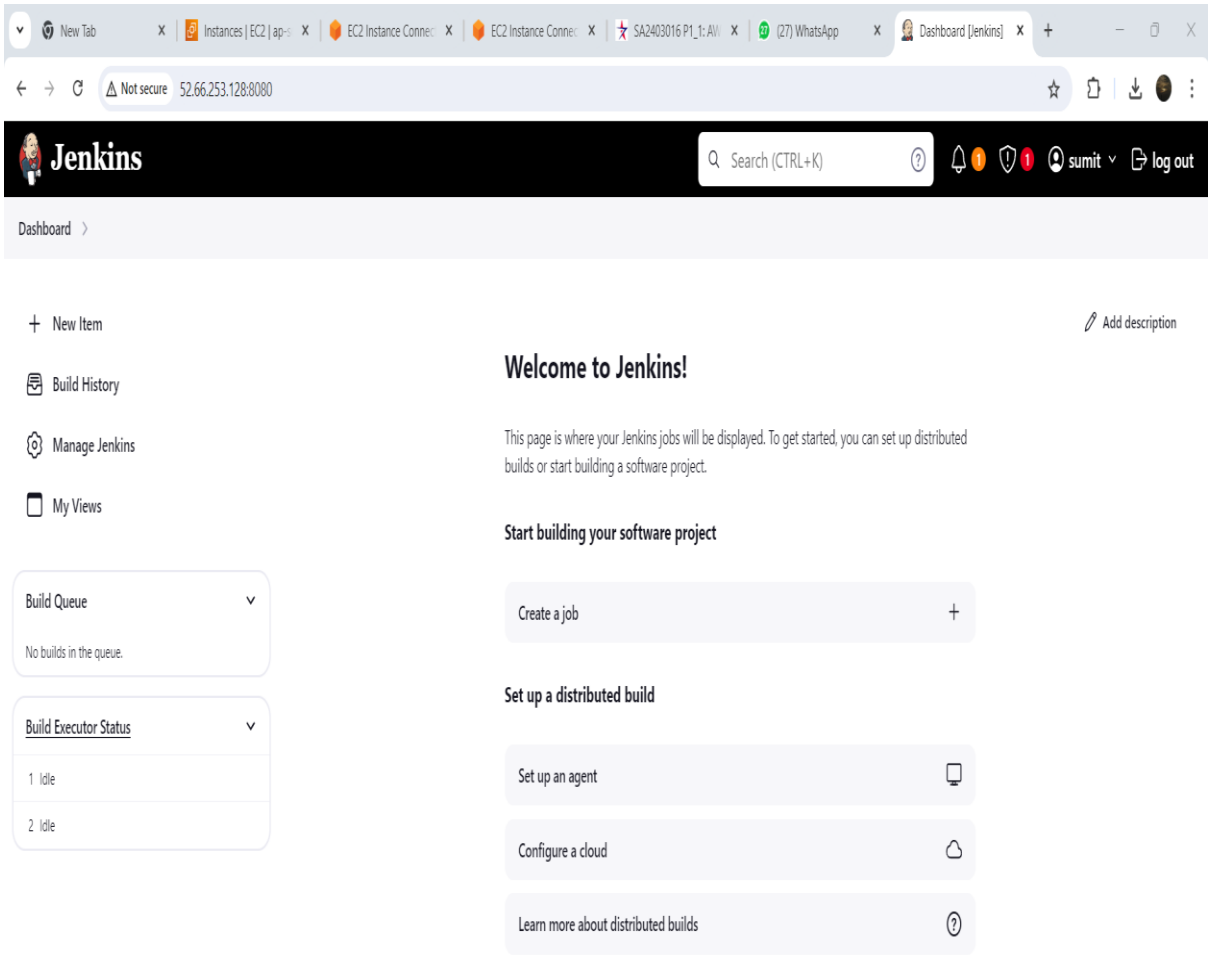
- **Now click on save and finish**



- **You can see now Jenkins is ready,now click on start using jenkins**

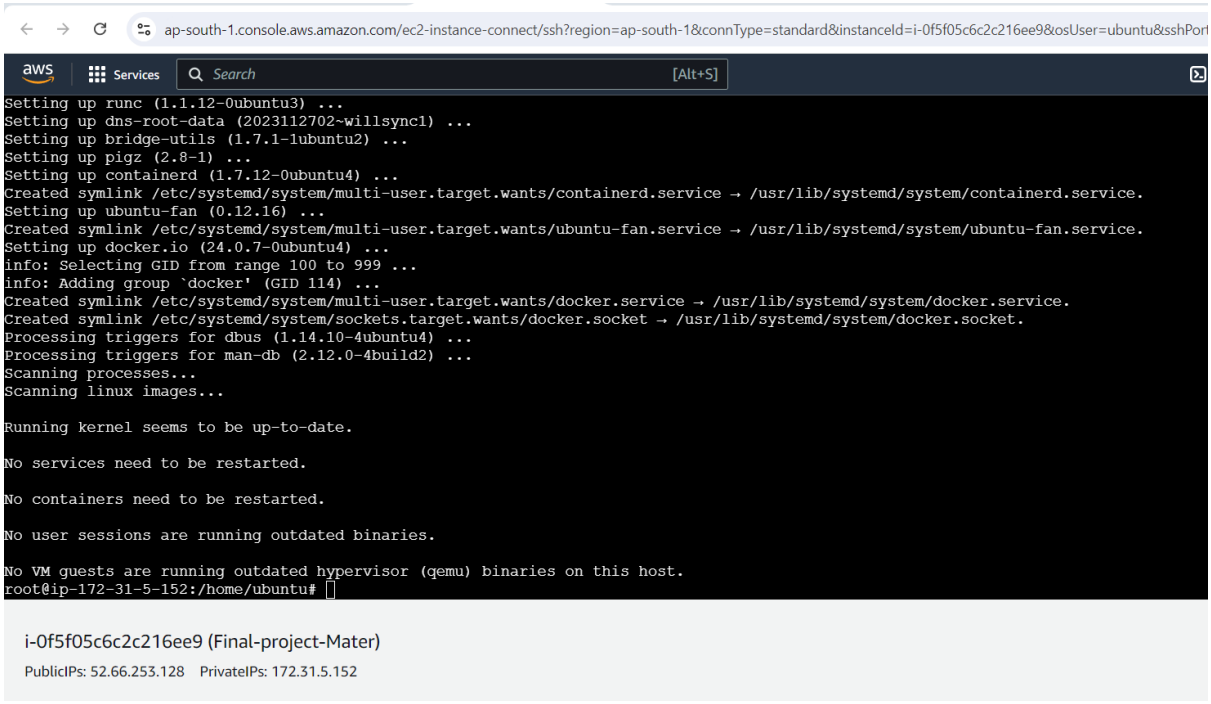


- **Now you can see the Jenkins home page**



Step 20: Now install docker in main machine

$ apt install docker.io -y

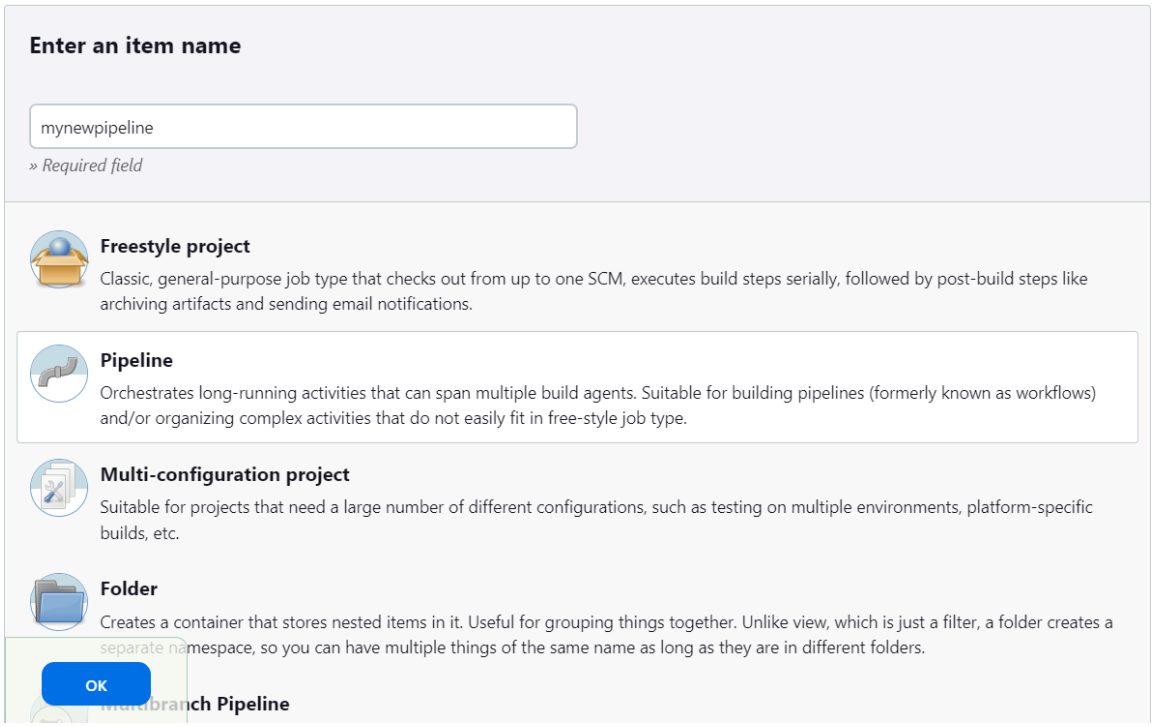


- **Now execute below command to give permission to docker then restart the jenkins**

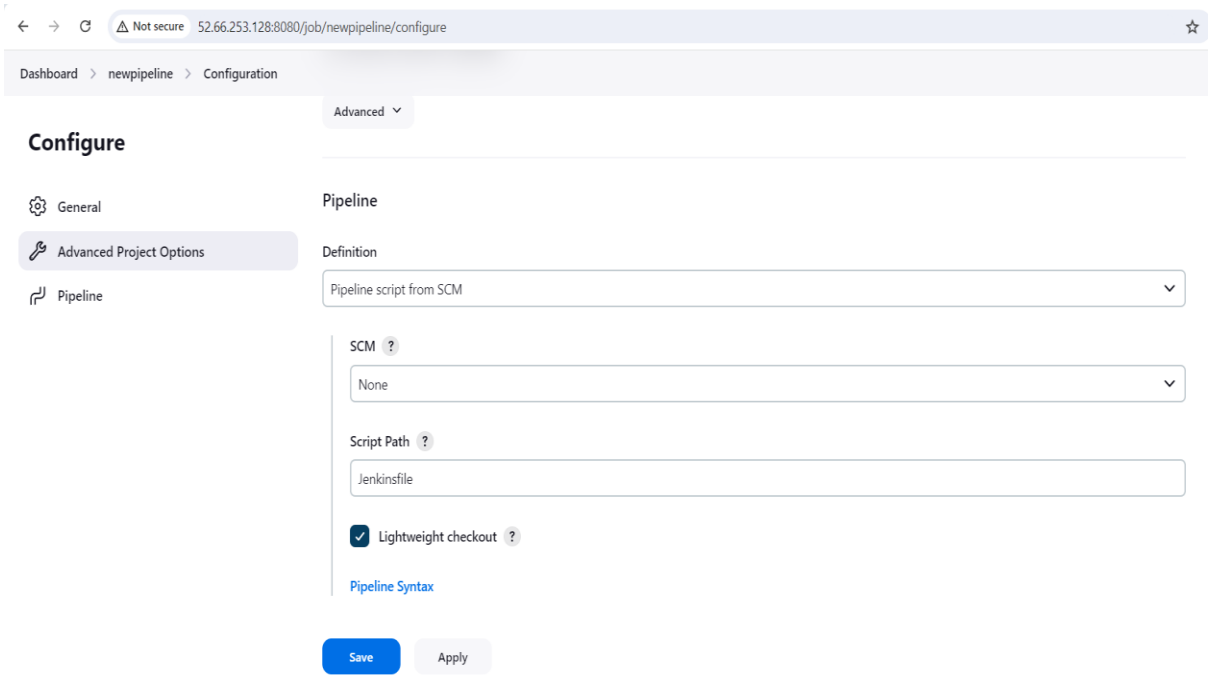
$usermod -aG docker jenkins

$service jenkins restart

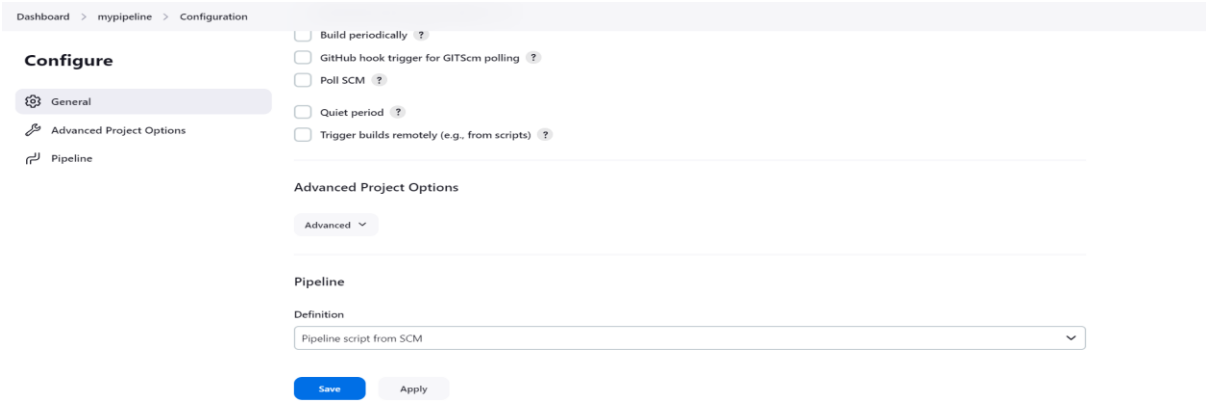
Step 21: Now reload the Jenkins and click on new item,and create a pipeline with name newpipelinen then click on ok



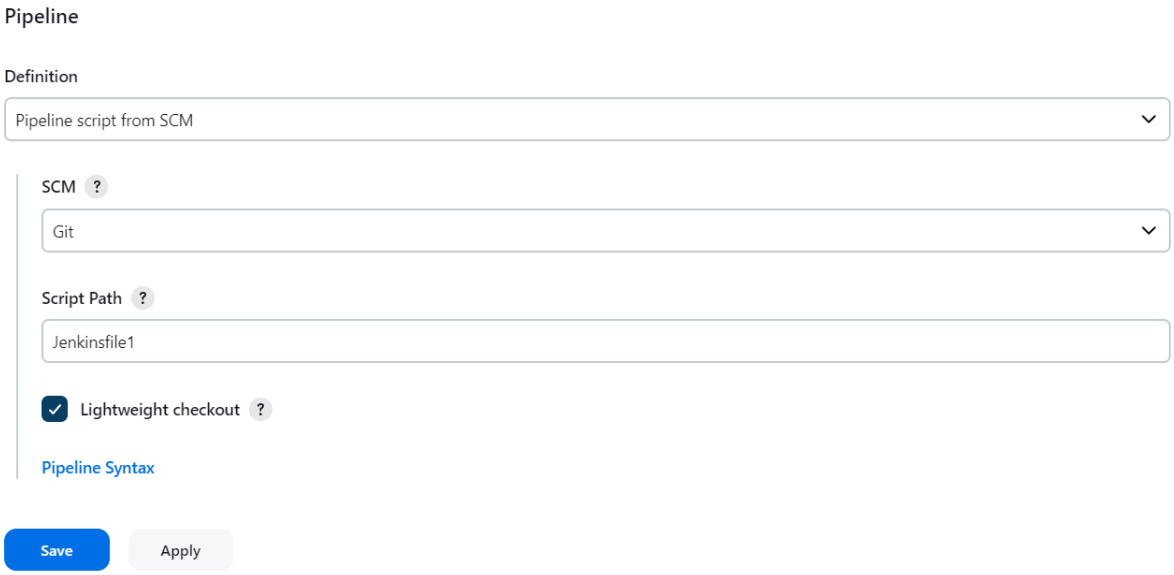
- **Now click on pipeline**



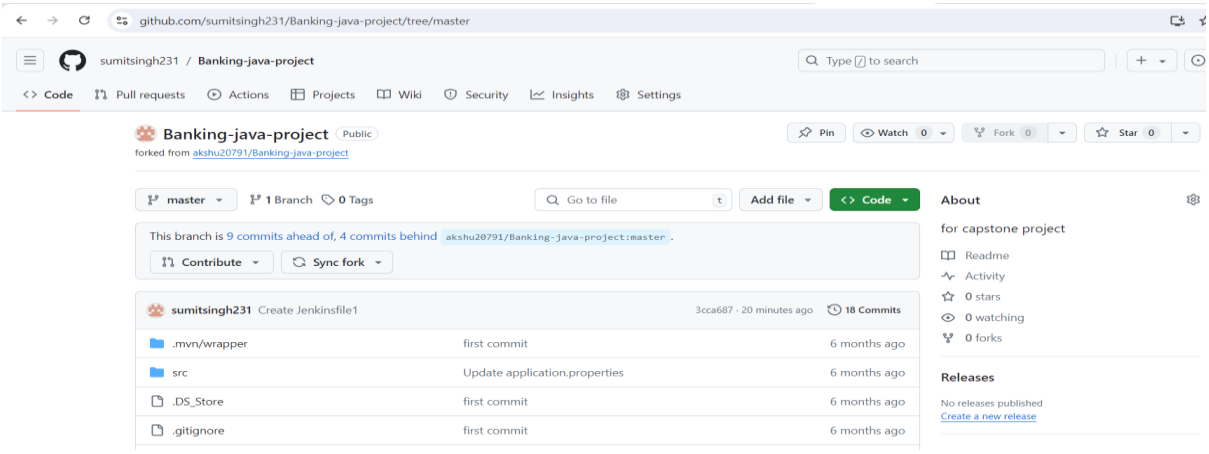
- **Now select pipeline script from SCM by dropdown**



- **Now select Git in SCM(source code management)**



Step 22: Now go to Github repository of Banking-java-project

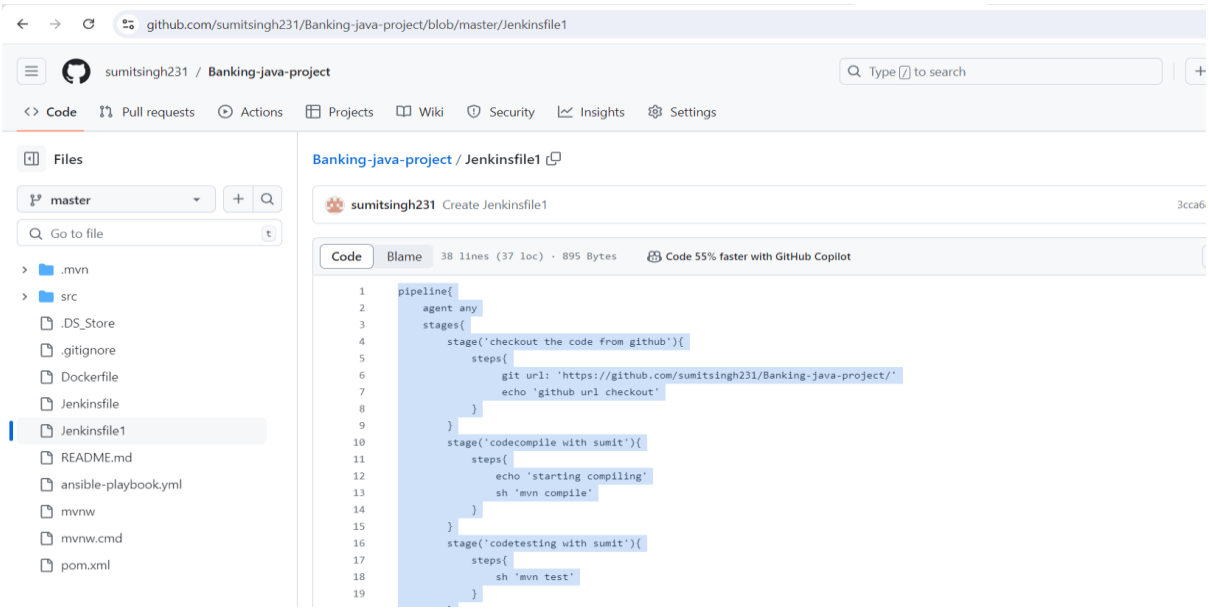


- There Create a new file with name jenkinsfile1 and pasted the below code
- Code describe the below points
  - Taking the banking project
  - Start code compiling
  - Code tesing
  - Package

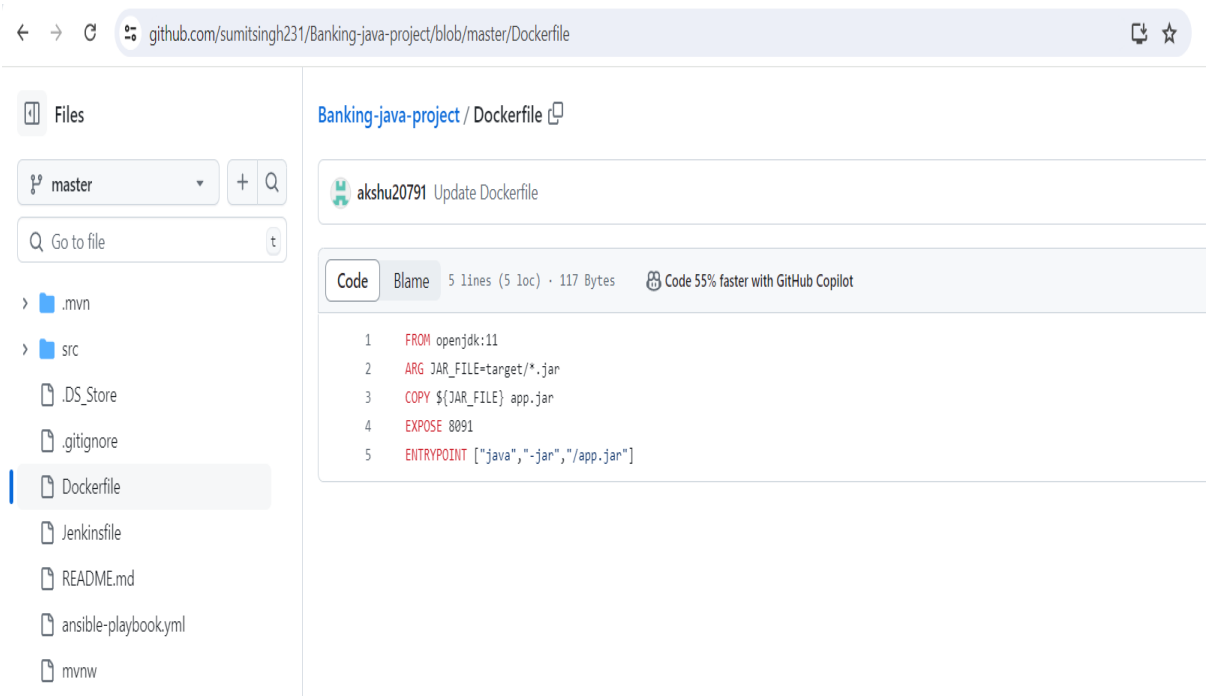
- Code-

```
pipeline{
  agent any
  stages{
    stage('checkout the code from github'){
      steps{
        git url: 'https://github.com/sumitsingh231/Banking-java-project/'
        echo 'github url checkout'
      }
    }
    stage('codecompile with sumit'){
      steps{
        echo 'starting compiling'
        sh 'mvn compile'
      }
    }
    stage('codetesting with sumit'){
      steps{
        sh 'mvn test'
      }
    }
    stage('qa with sumit'){
```

```
        steps{
            sh 'mvn checkstyle:checkstyle'
        }
    }
    stage('package with sumit'){
        steps{
            sh 'mvn package'
        }
    }
    stage('run dockerfile'){
     steps{
        sh 'docker build -t myimg .'
      }
     }

  }
}
```

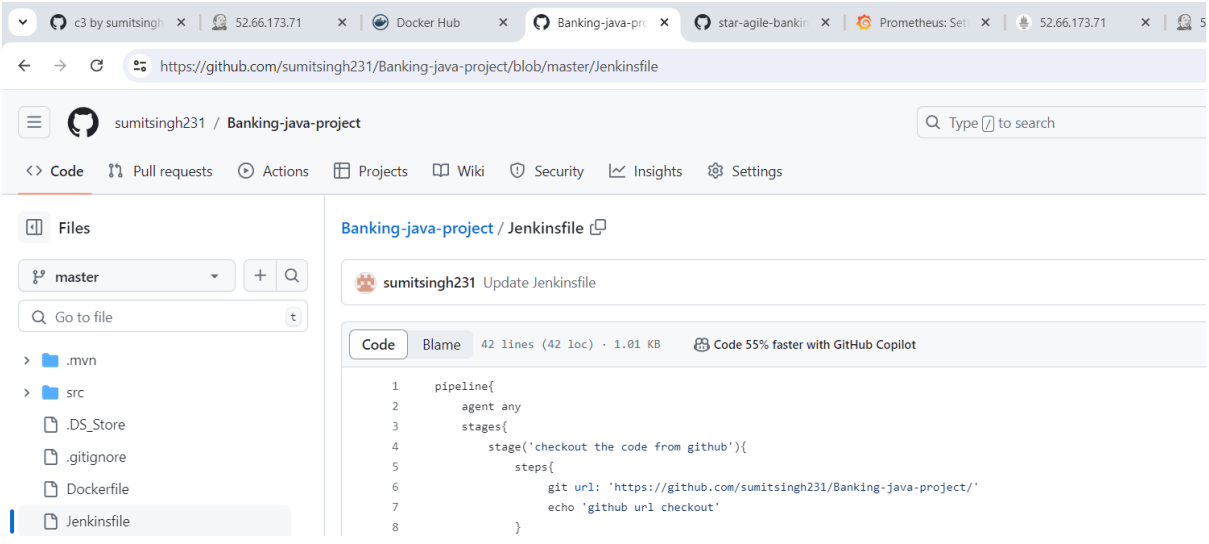


Step 23: Now click on dockerfile in repository and edit export port 8091

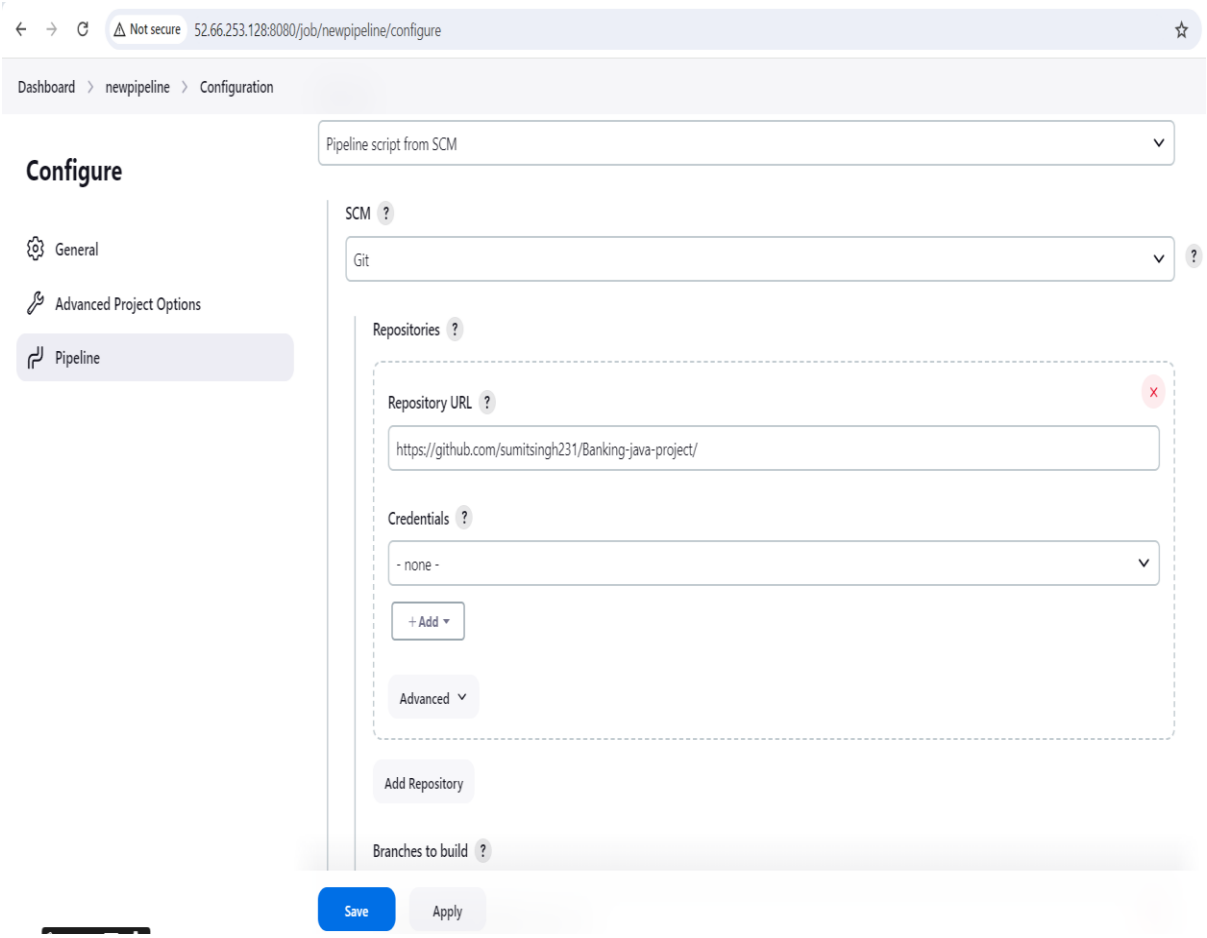


Step 24: Copy the repository link

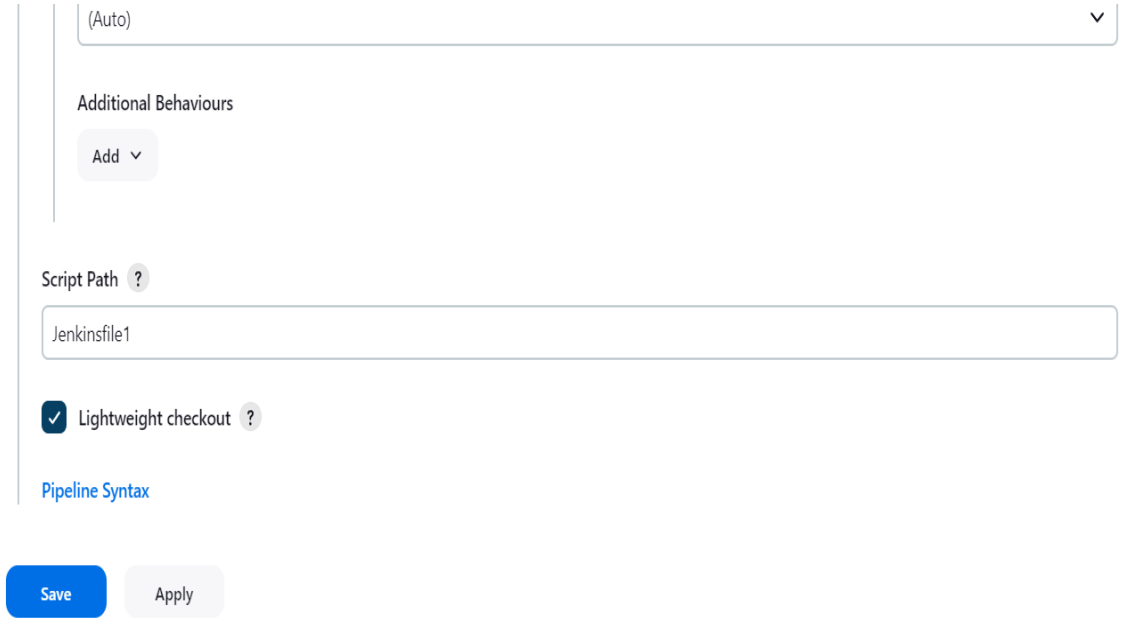
https://github.com/sumitsingh231/Banking-java-project/



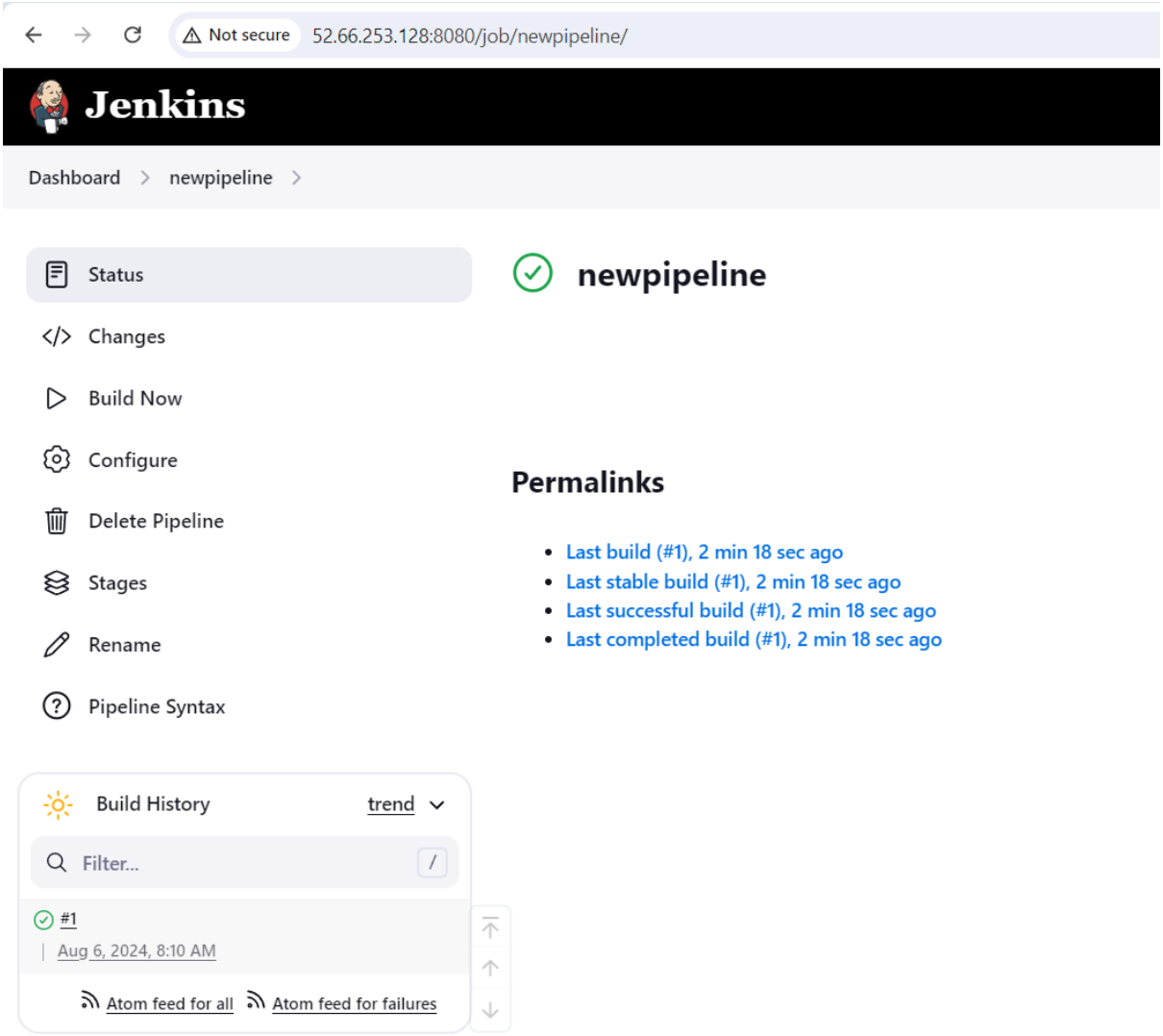
- **Now come back to the Jenkins page and Paste this link into the Jenkins page**



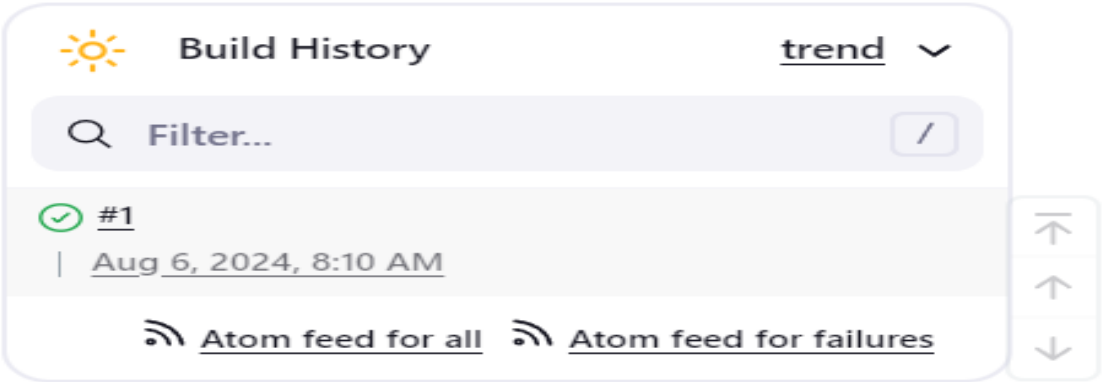
- **Now click on save**



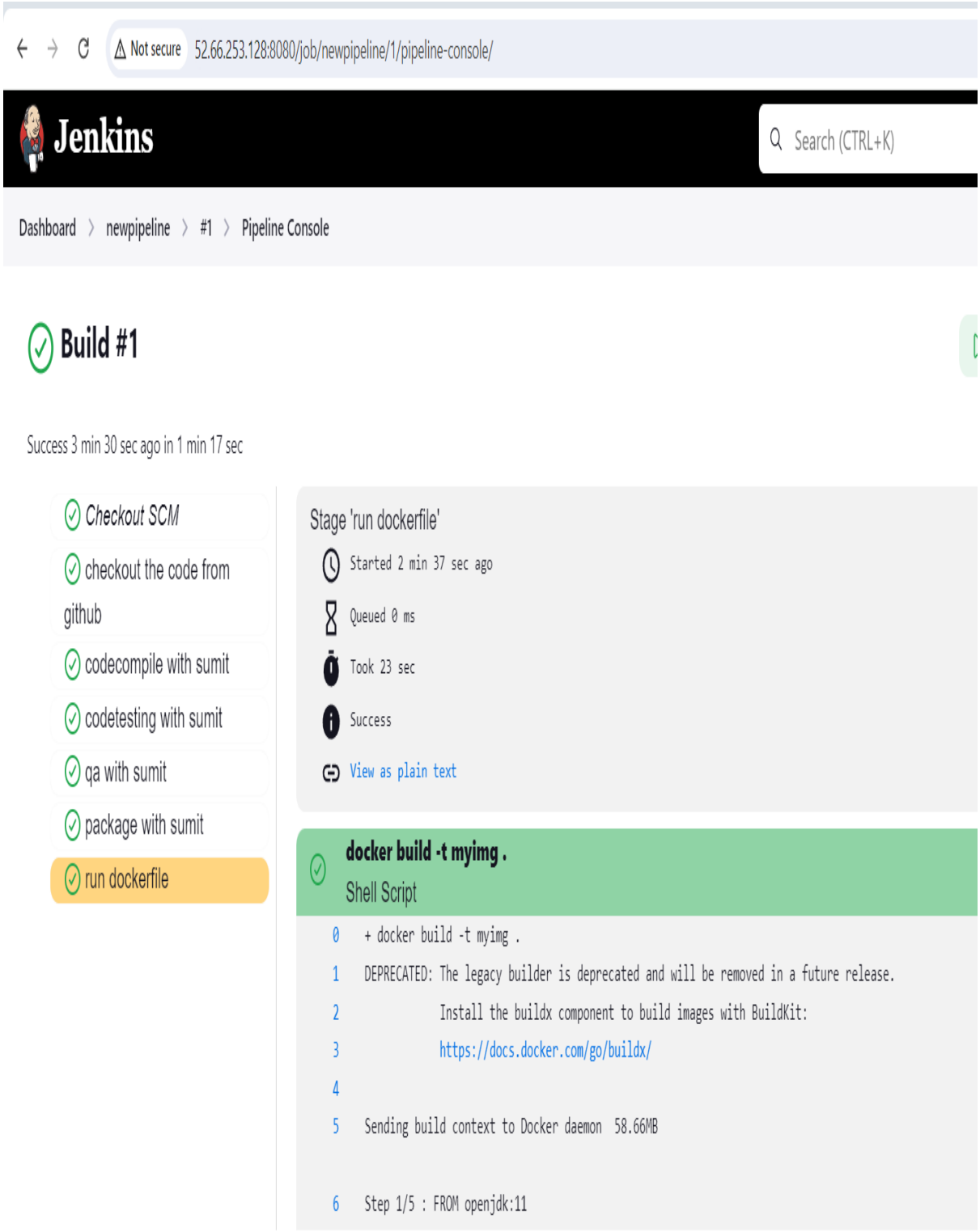
Step 25: Now click on build now,You can see in build history build get executed successfully



- **Now click on #1 (Build) to see the build**



- **Now click on pipeline console**

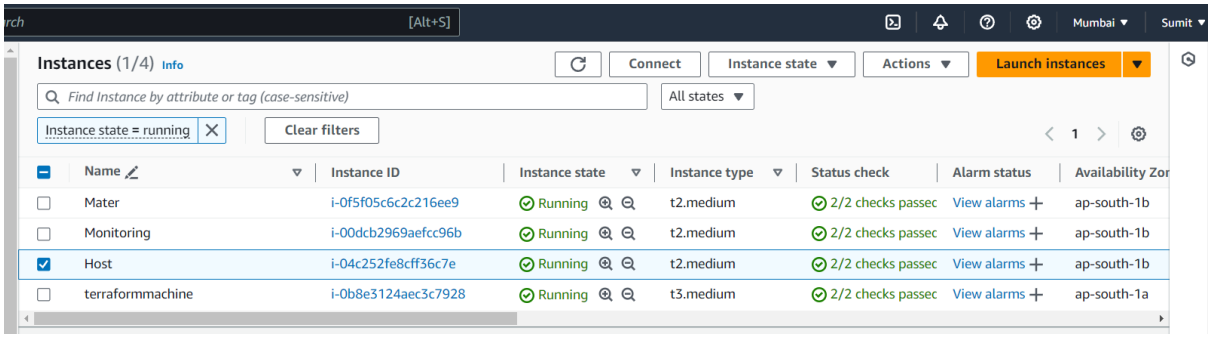


- **Here you can see the console ouput**

- **Now click on the pipeline overview,here you can see the execution in graph format**

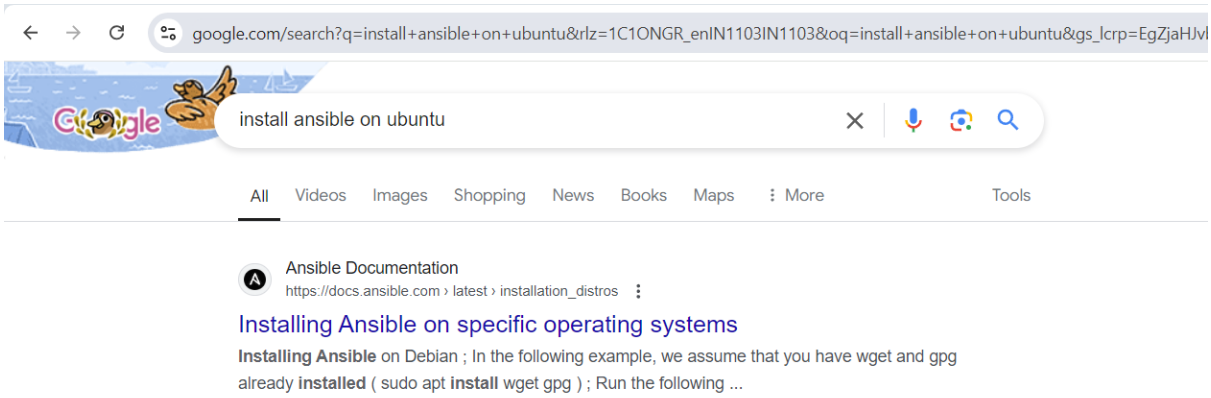
- **You can see I have build the image**
- **In above pic you can see the build steps**
  - **Start the build**
  - **Checkout the code from Github**
  - **Code compile with sumit**
  - **Code testing with sumit**
  - **QA with sumit**

- **Now renaming our instance as Master,Host and Monitoring**



Step 26: Now install ansible inMaster instance

- **Browse install ansible on ubuntu and click on ansible documentation**



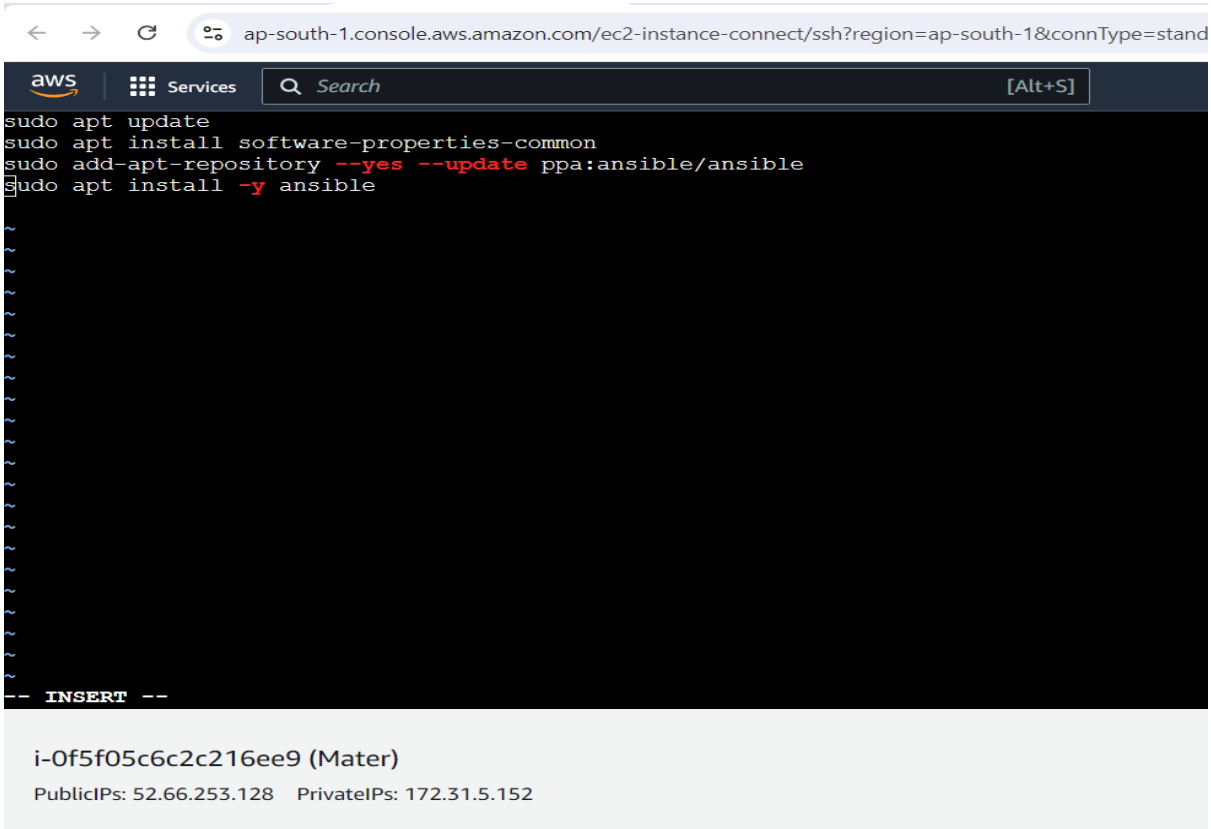
- **Scroll down and click on Installing Ansible on ubuntu**

- **Copy all commands**

sudo apt update
sudo apt install software-properties-common
sudo add-apt-repository --yes --update ppa:ansible/ansible
sudo apt install ansible

- **Create a file with name ansible.sh and paste copied command**

$ vi ansible.sh



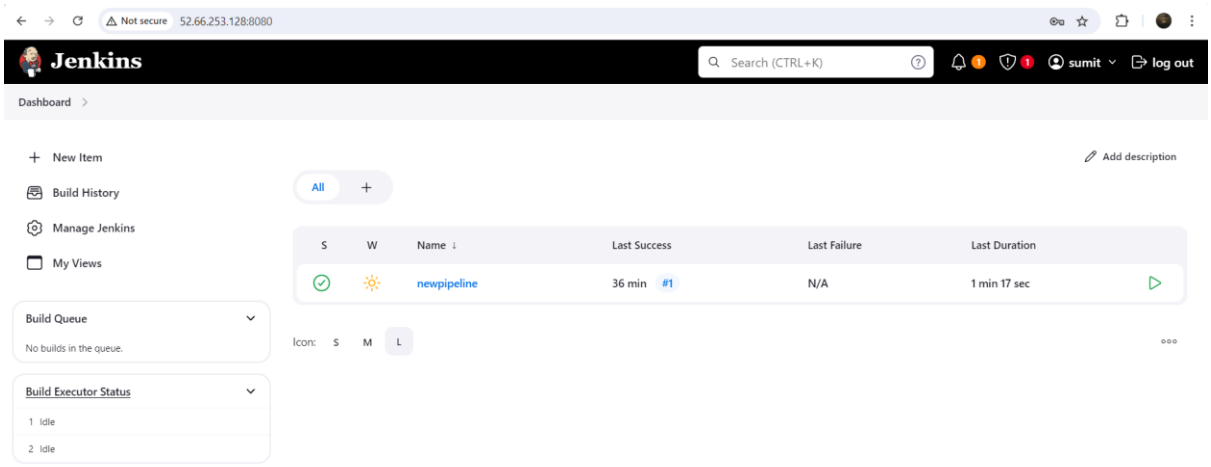
Step 27: Now Execute below command to run script file

$sh ansible.sh

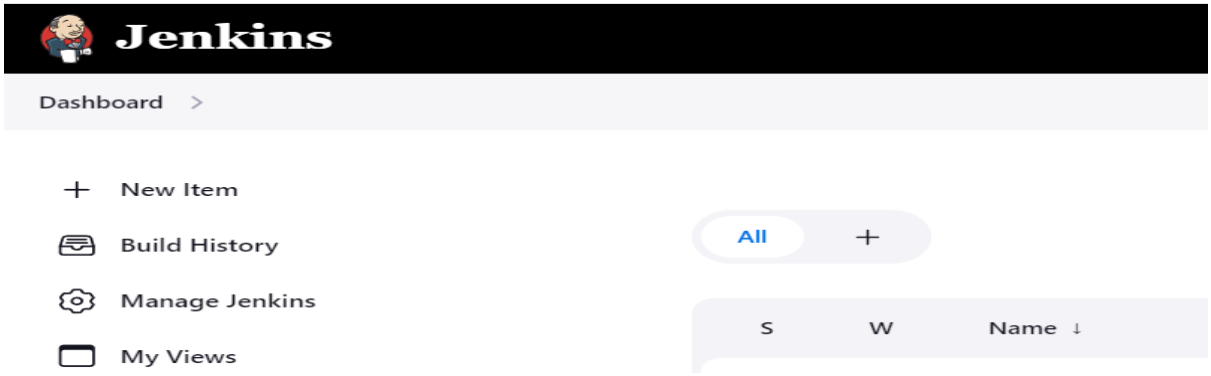
- **You can see we have installed ansible in master machine**

Step 28: Now gain restart the Jenkins and refresh the Jenkins page

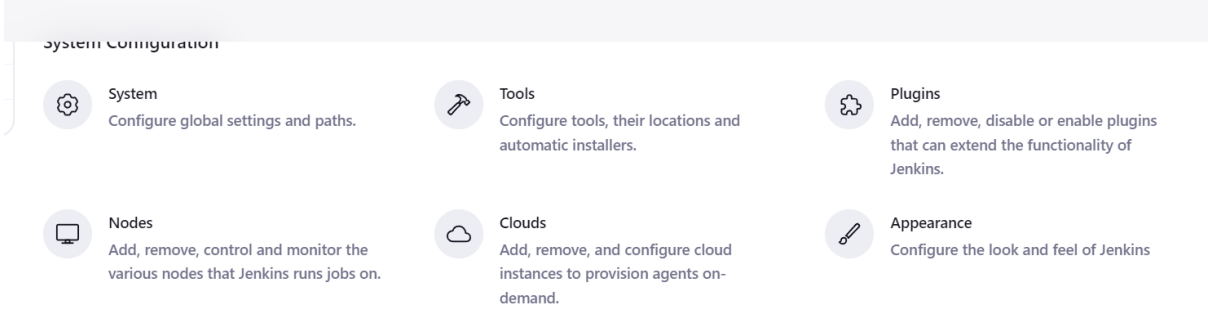
$service jenkins restart



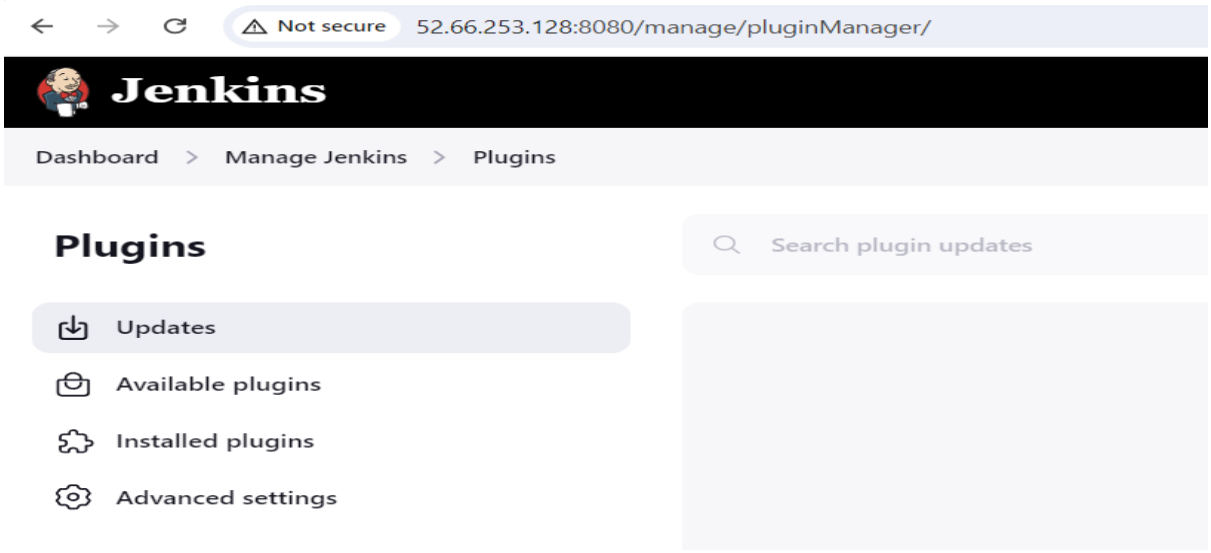
- **Click on manage Jenkins**



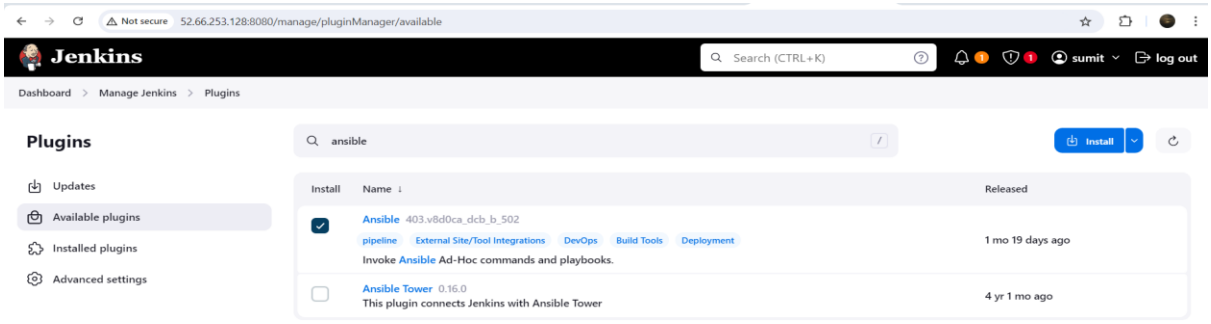
- **Click on plugins**



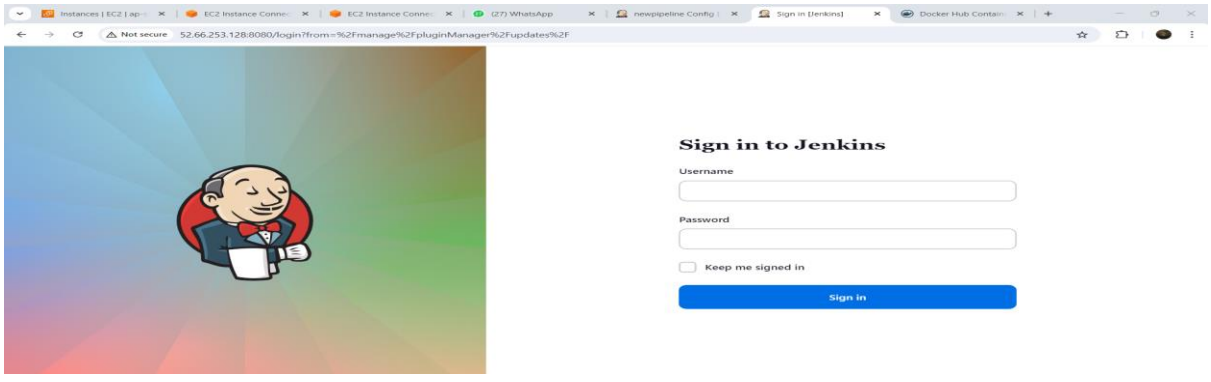
- **Click on Available plugins**



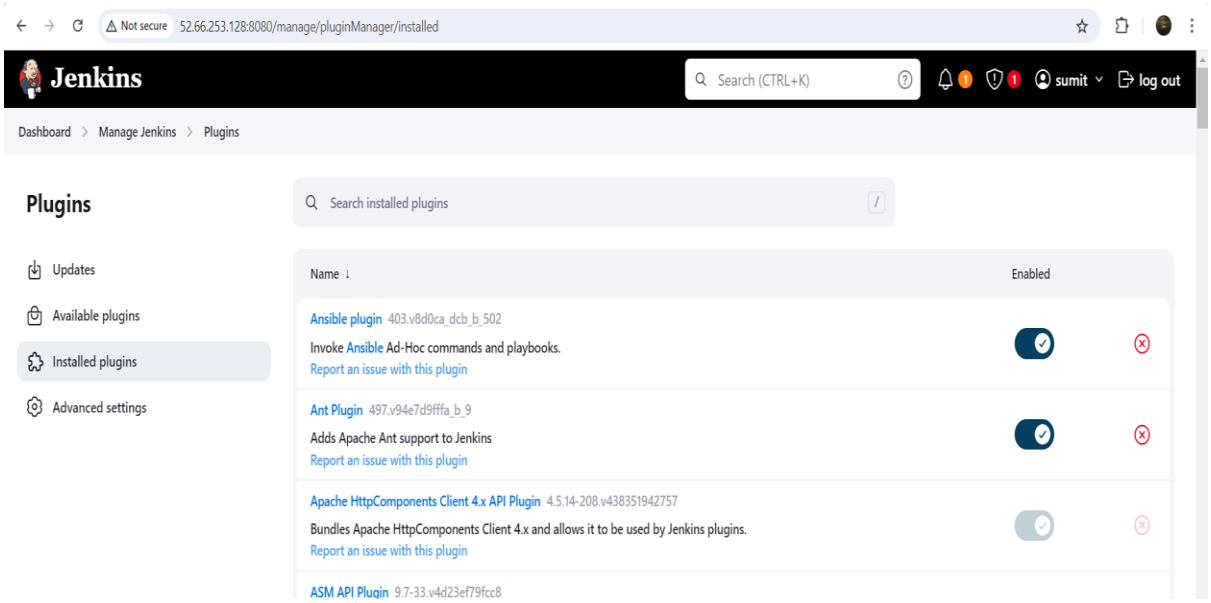
- **Type ansible for search and select ansible plugin and click on install and select restart option**



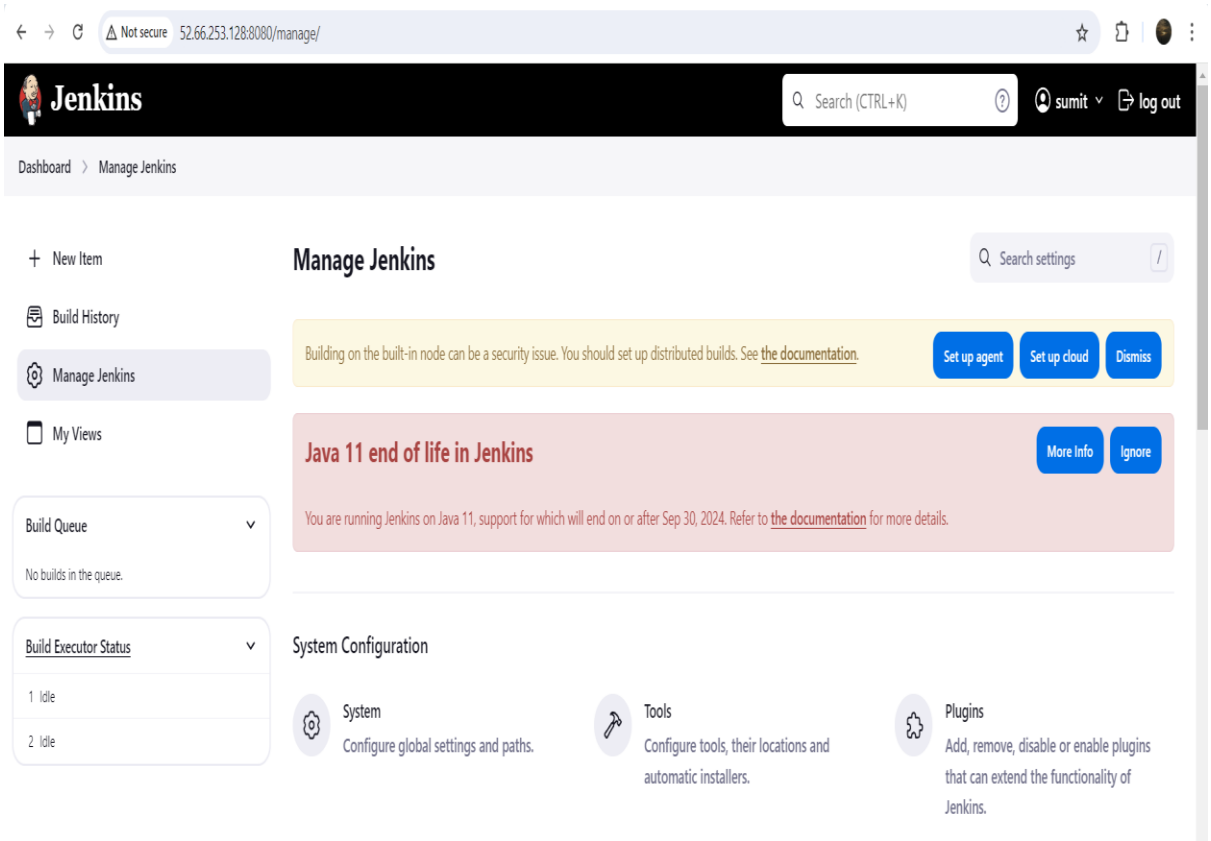
- **You can see Jenkins restarted**



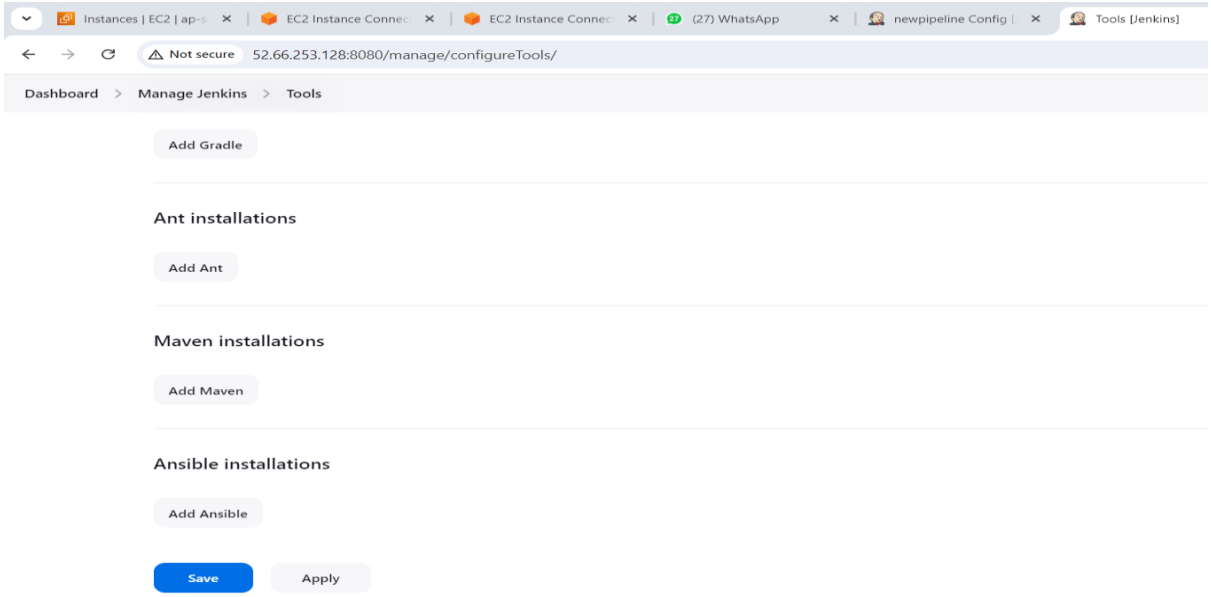
- **Login again and click on installed plugin you will see there is ansible plugin which we have installed**



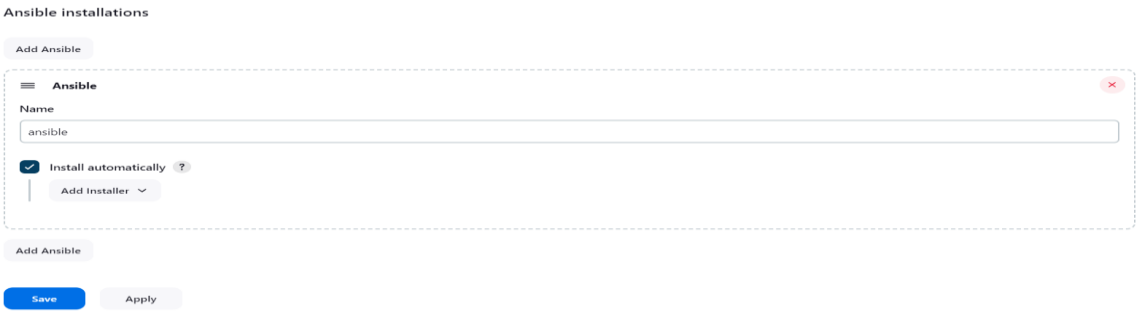
- **Now click on manage Jenkins and go to tools option for configure the ansible tool**



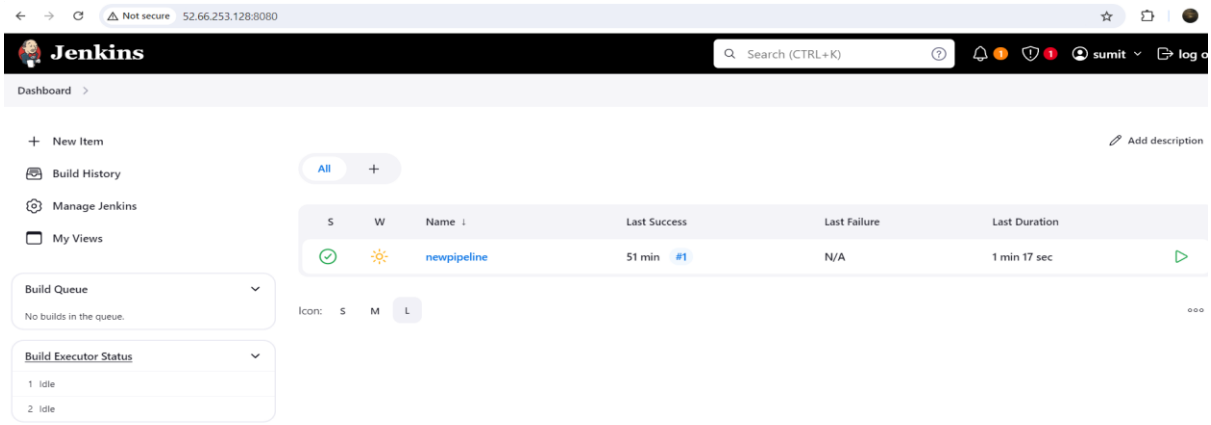
- **Scroll down and click on add ansible**



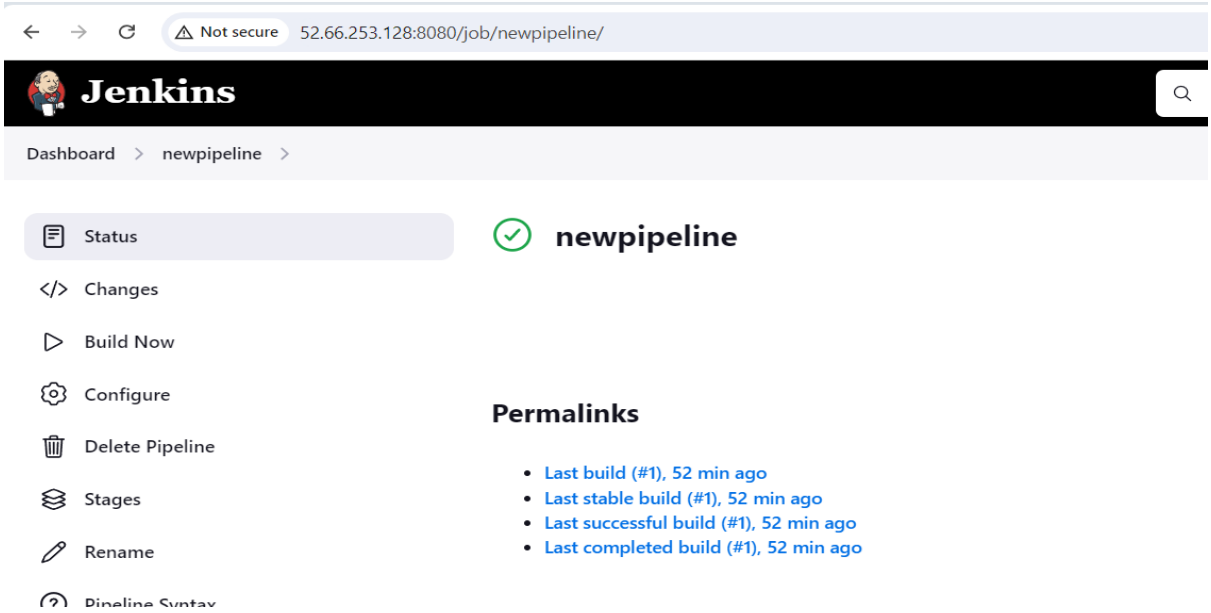
- **Type name as ansible and tick on install automatically,then click first on apply then save**



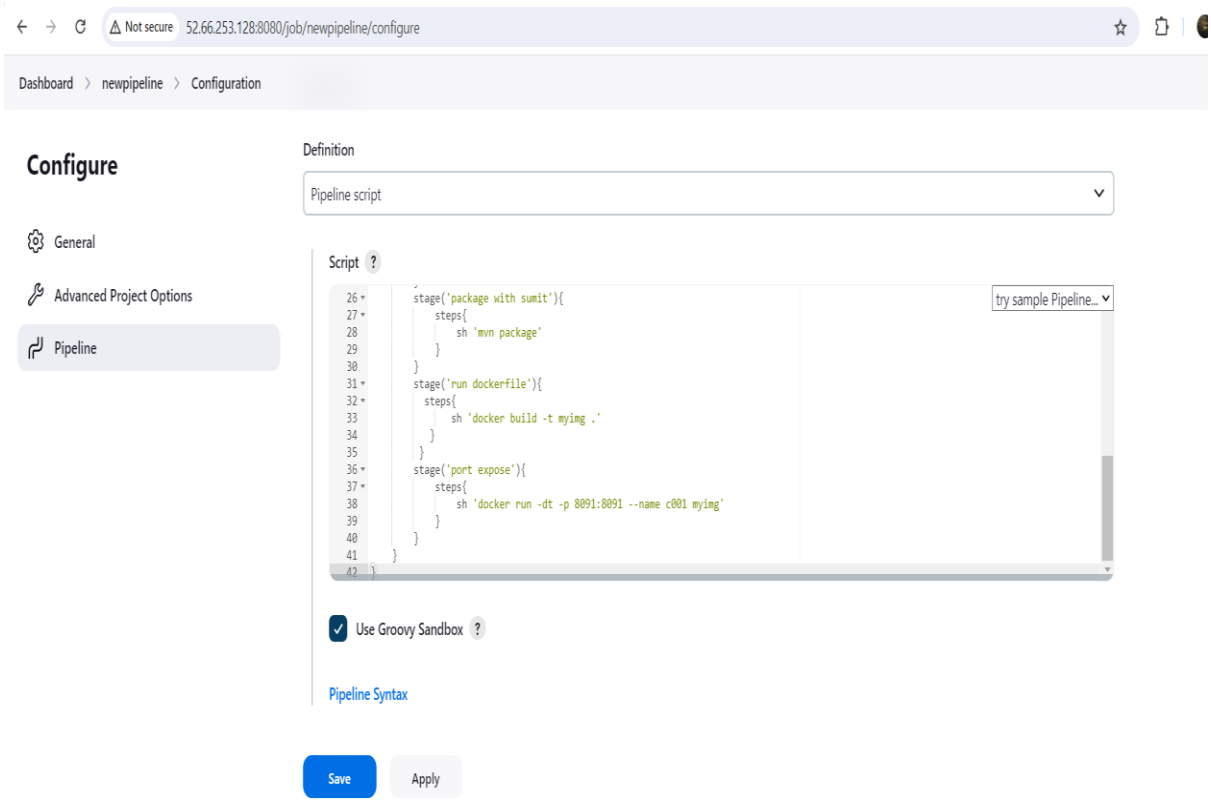
- **Now go to the dashboard**



- **Click on newpipeline**



- **Now go to configure and in pipeline script paste the code from Jenkins file**



- **Modify the code**

```
27          steps{
28              sh 'mvn package'
29          }
30      }
31      stage('run dockerfile'){
32        steps{
33            sh 'docker build -t sumitsingh231/myproject:1 .'
34          }
35        }
36      stage('Login the docker hub and push the file'){
37          steps{
38
39          }
40      }
41  }
42  }
```

- **Now click on pipeline syntex and select plugin**

Overview

This **Snippet Generator** will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in from the list, configure it, click **Generate Pipeline Script**, and you will see a Pipeline Script statement that would call the step with that configuration. You may copy and paste the whole statement into your script or pick up just the options you care about. (Most parameters are optional and can be omitted in your script, leaving them at default values.)

Steps

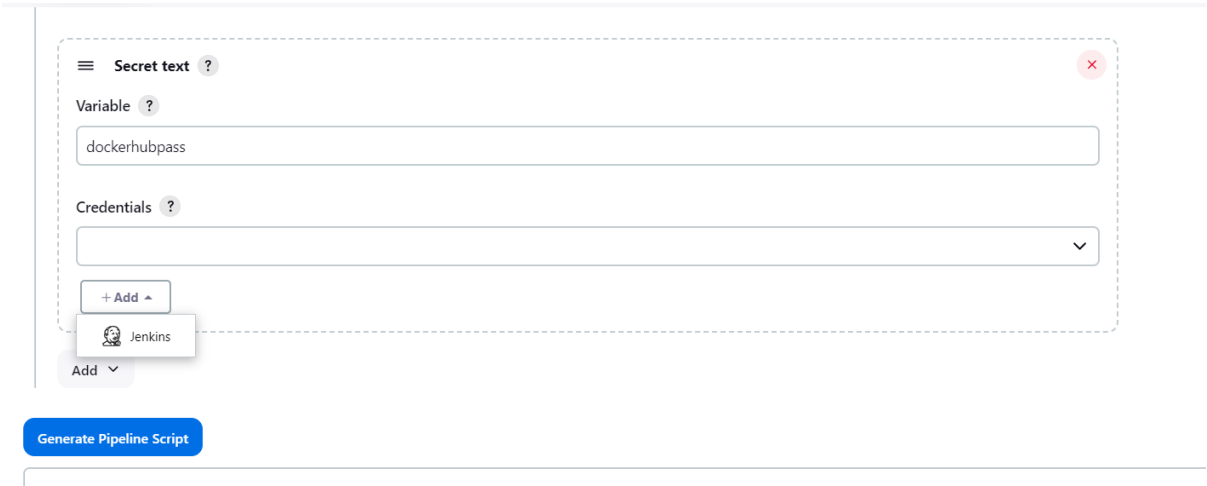
Sample Step

withCredentials: Bind credentials to variables

withCredentials ?

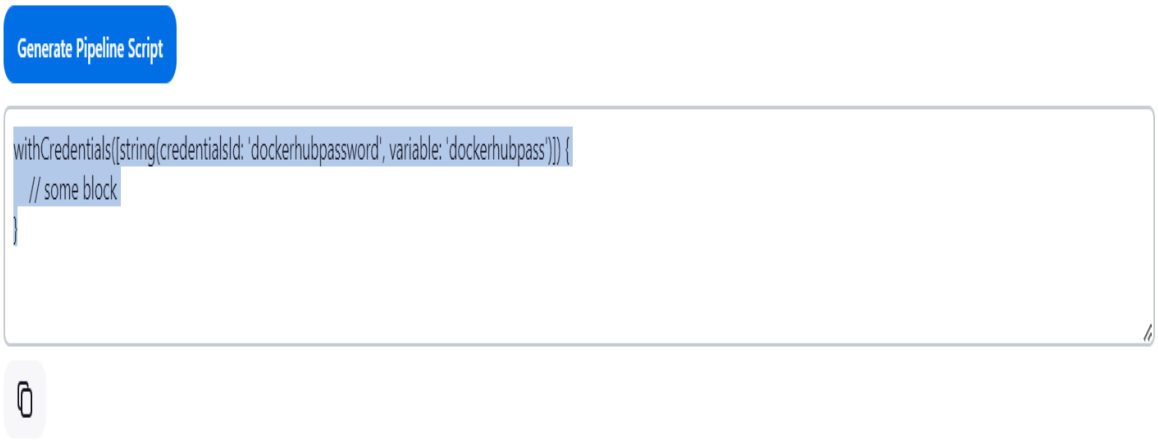
Secret values are masked on a best-effort basis to prevent *accidental* disclosure. Multiline secrets, such as the contents of a SSH private key file, are not masked. See the inline help for details and usage guidelines.

- **Click on add then secret text,give name of variable as dockerpass,then click on add and select jenkins**



- **Select secret text in kind**
- **In secret giving dockerhub account password**
- **In ID typing dockerhubpassword**
- **In description dockerhubpass by sumit**
- **Then click on add**

- **Now click on generate pipeline script,copy it**



Global Variables

- **Now go to pipeline syntex**
- **Type code as below,then click on apply then save**

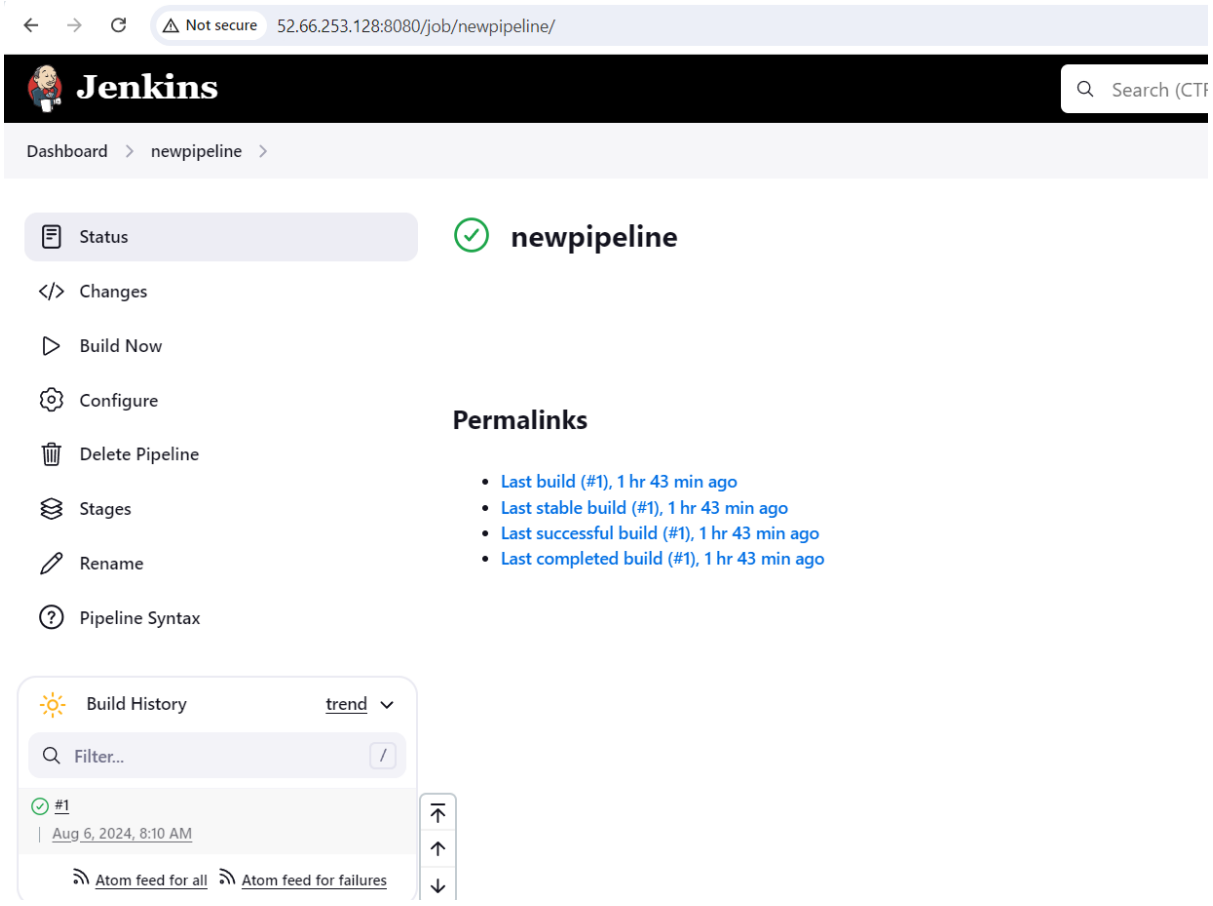
```
        }
    }
    stage('run dockerfile'){
      steps{
          sh 'docker build -t sumitsingh231/myproject:1 .'
      }
    }
    stage('Login the docker hub and push the file'){
        steps{
            withCredentials([string(credentialsId: 'dockerhubpassword', variable: 'dockerhubpass')]) {
                sh 'docker login -u sumitsingh231 -p ${dockerhubpass}'
}
        }
    }
  }
}
```

Use Groovy Sandbox ?

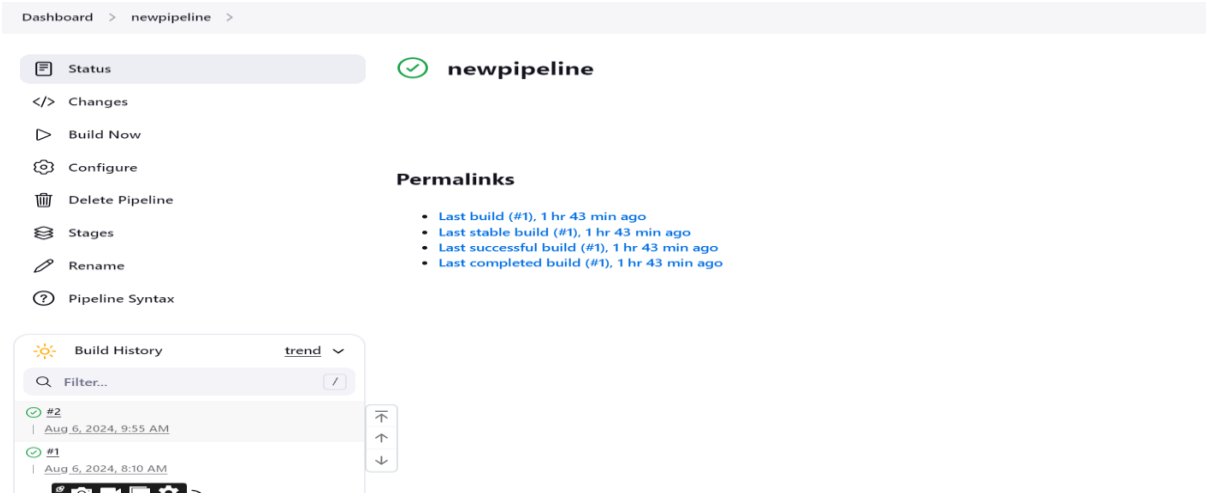
Pipeline Syntax

Save Apply

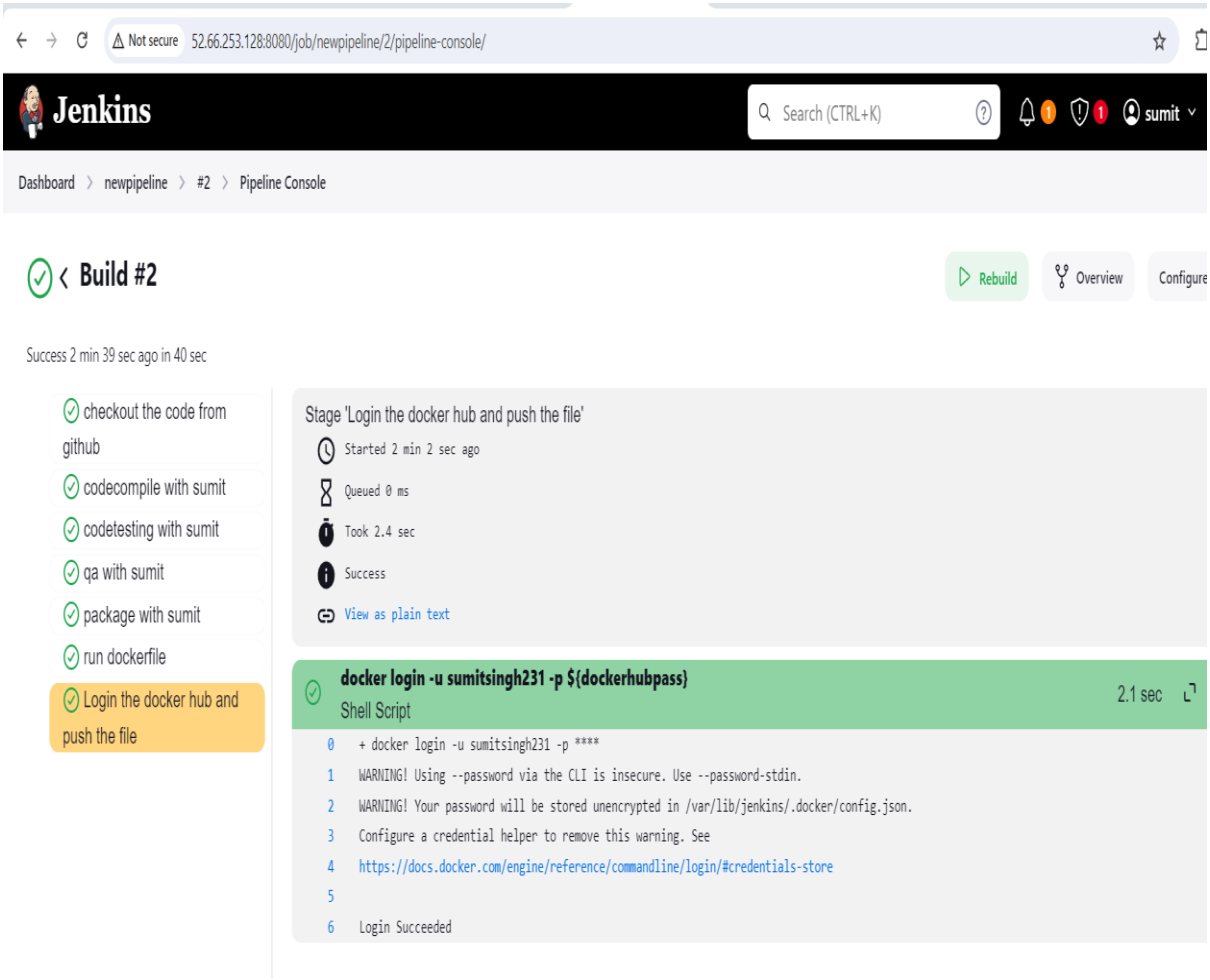
- Now click on build now



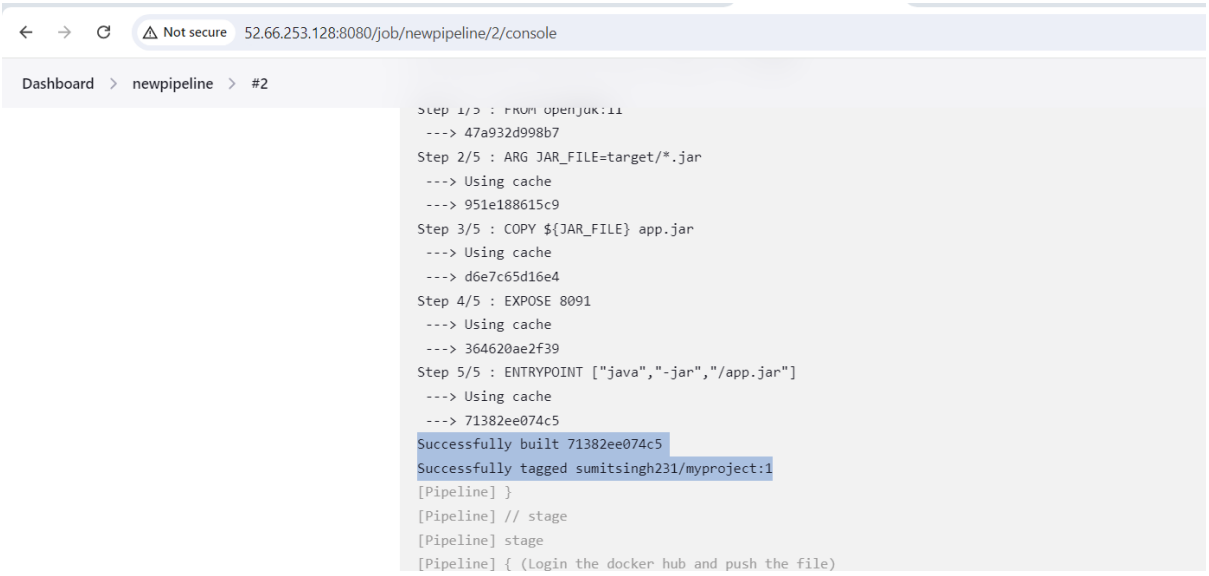
- You can see build success

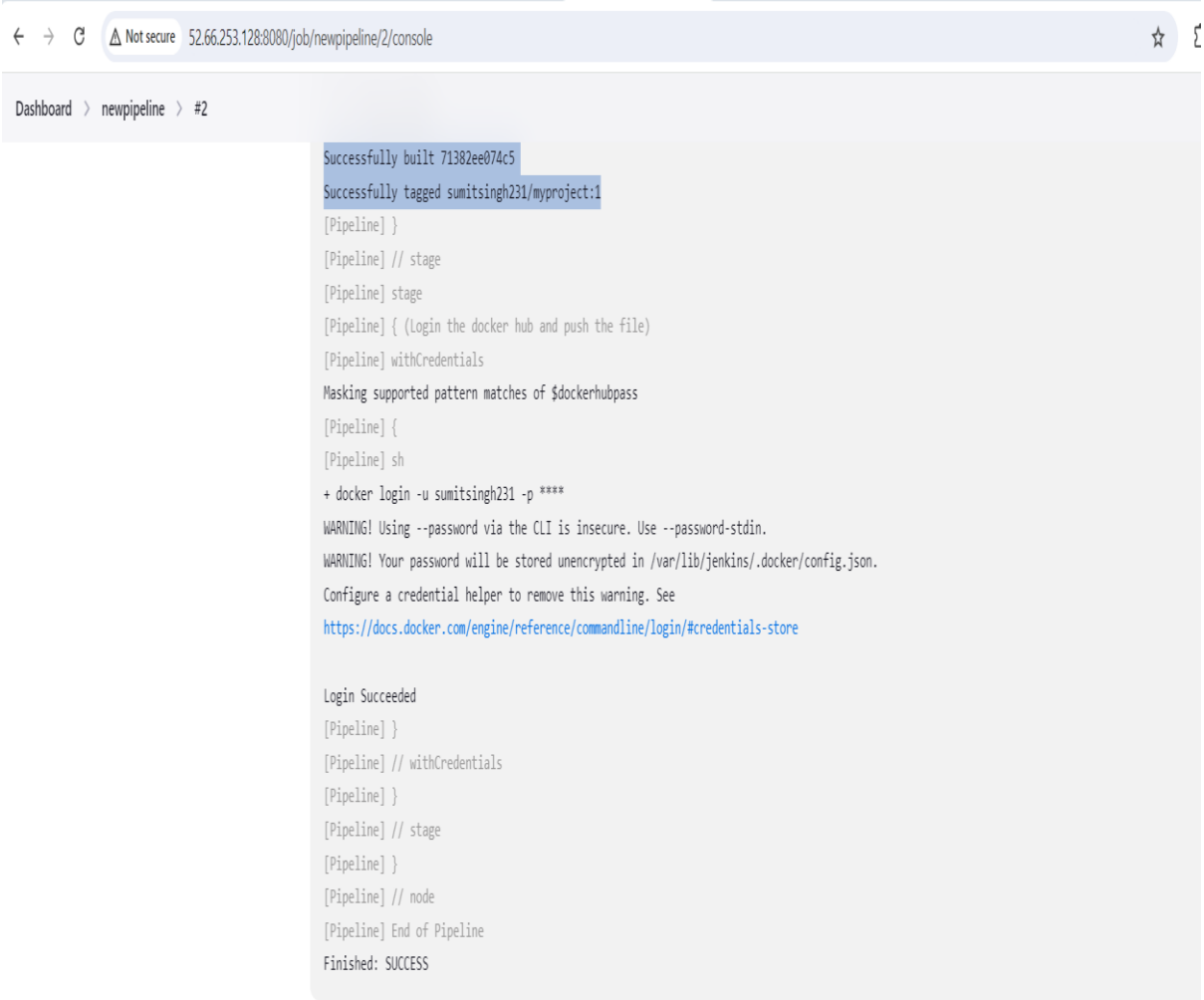


- Go to the pipeline console,you can see it will not shown the password

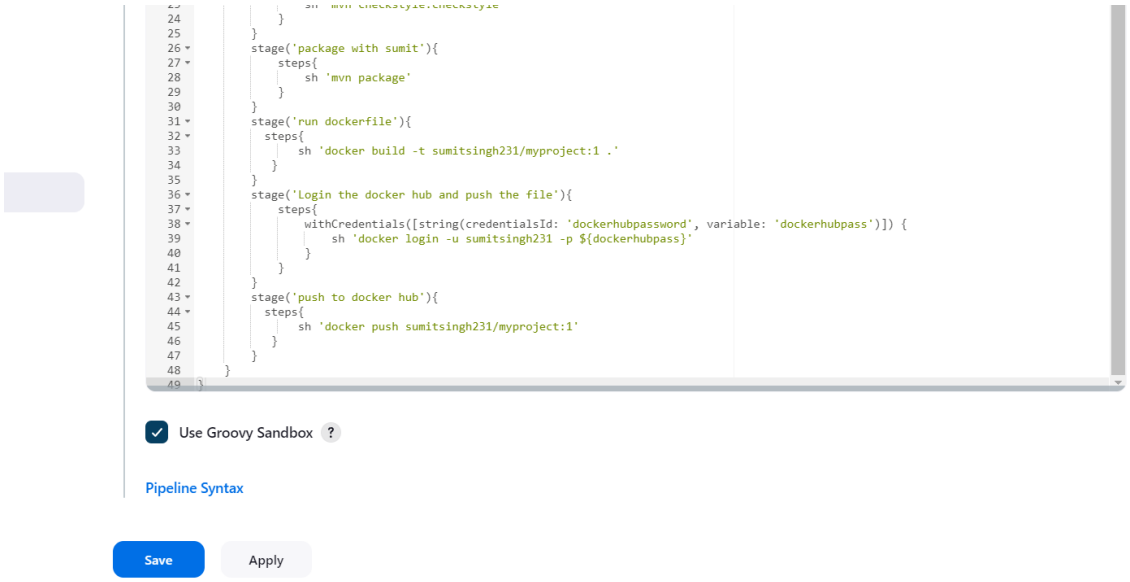


- **In console output you can see it has build as well as tagged successfully**





- **Now again go to the configure and modify the file for push the image to docker hub,then click on apply then save**



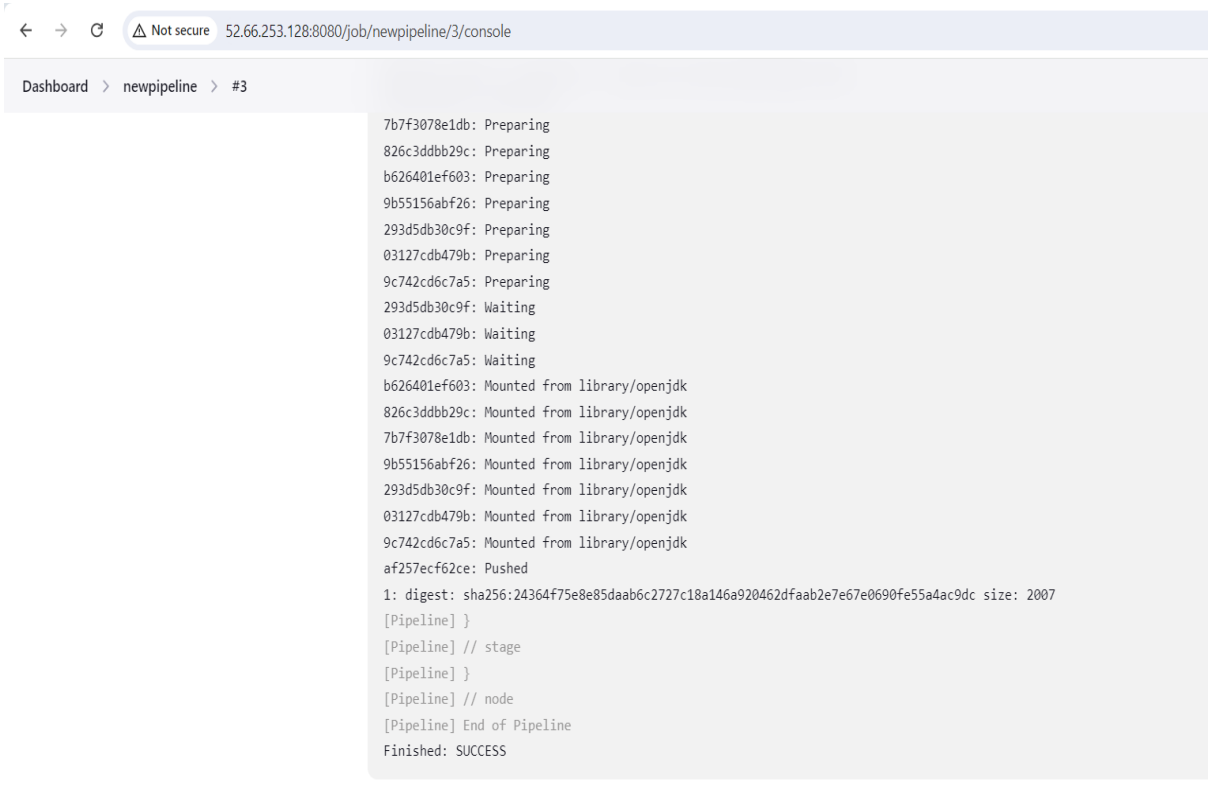
```
        }
    }
    stage('package with sumit'){
        steps{
            sh 'mvn package'
        }
    }
    stage('run dockerfile'){
      steps{
          sh 'docker build -t sumitsingh231/myproject:1 .'
       }
    }
    stage('Login the docker hub and push the file'){
        steps{
            withCredentials([string(credentialsId: 'dockerhubpassword', variable: 'dockerhubpass')]) {
                sh 'docker login -u sumitsingh231 -p ${dockerhubpass}'
            }
        }
    }
    stage('push to docker hub'){
      steps{
          sh 'docker push sumitsingh231/myproject:1'
       }
    }
  }
}
```

Use Groovy Sandbox ?

Pipeline Syntax

Save Apply

- **Now again click on build now,you can see build is success in console output**

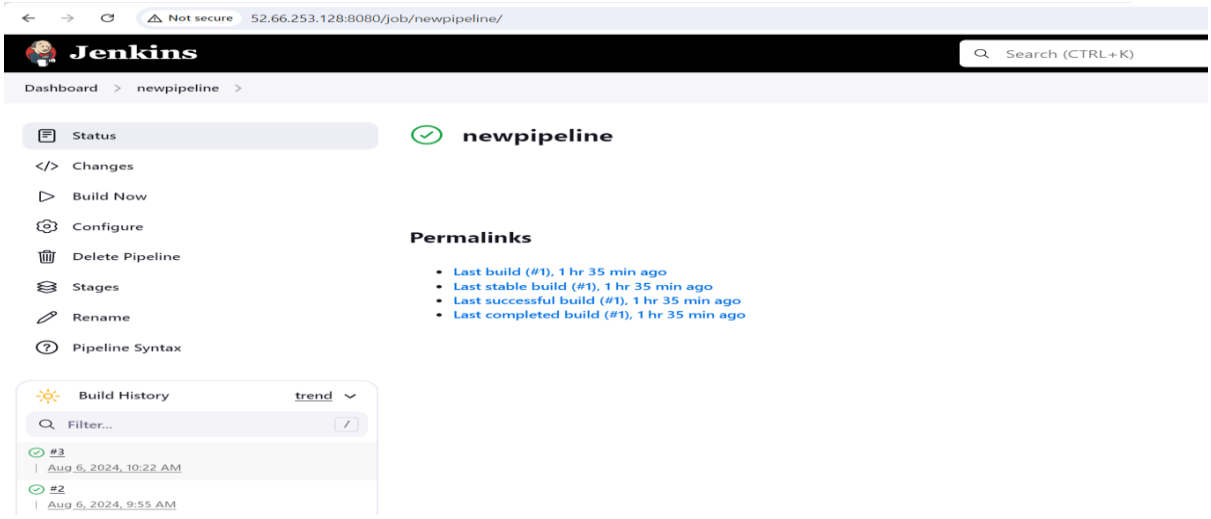


Not secure 52.66.253.128:8080/job/newpipeline/3/console

Dashboard > newpipeline > #3

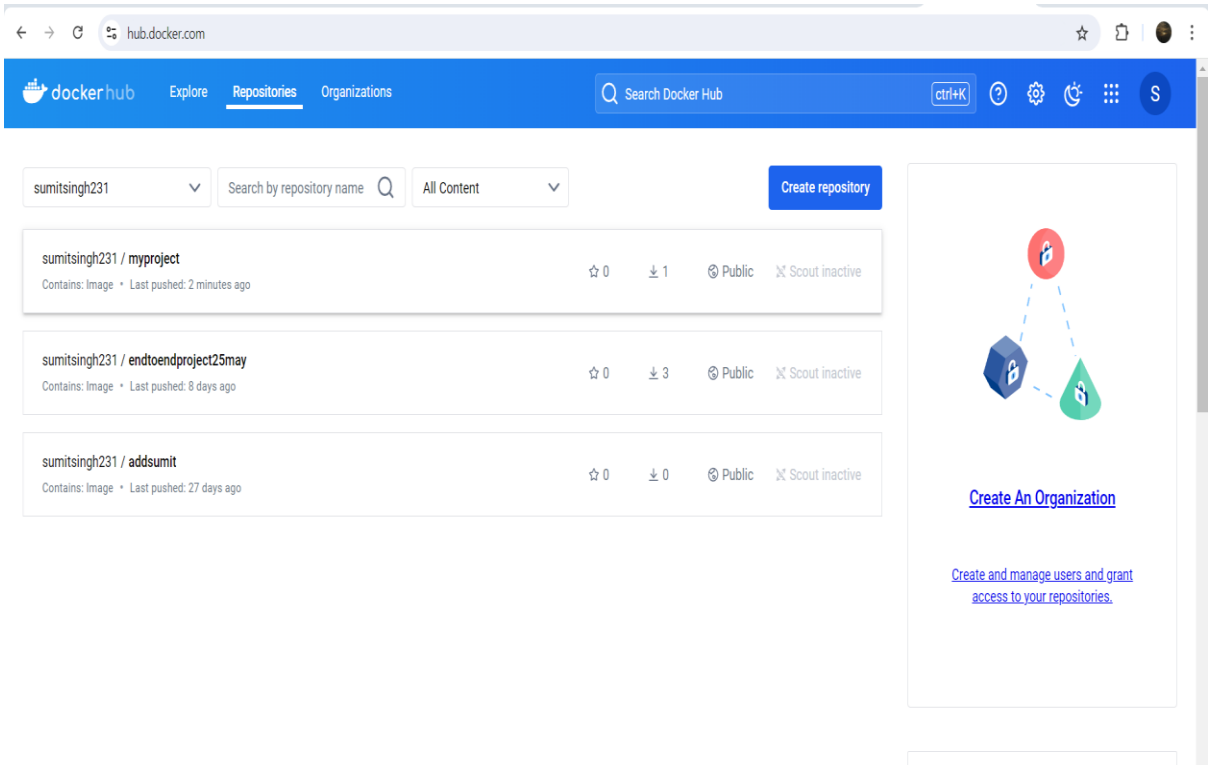
```
7b7f3078e1db: Preparing
826c3ddbb29c: Preparing
b626401ef603: Preparing
9b55156abf26: Preparing
293d5db30c9f: Preparing
03127cdb479b: Preparing
9c742cd6c7a5: Preparing
293d5db30c9f: Waiting
03127cdb479b: Waiting
9c742cd6c7a5: Waiting
b626401ef603: Mounted from library/openjdk
826c3ddbb29c: Mounted from library/openjdk
7b7f3078e1db: Mounted from library/openjdk
9b55156abf26: Mounted from library/openjdk
293d5db30c9f: Mounted from library/openjdk
03127cdb479b: Mounted from library/openjdk
9c742cd6c7a5: Mounted from library/openjdk
af257ecf62ce: Pushed
1: digest: sha256:24364f75e8e85daab6c2727c18a146a920462dfaab2e7e67e0690fe55a4ac9dc size: 2007
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

- **Here also you can see build got success**

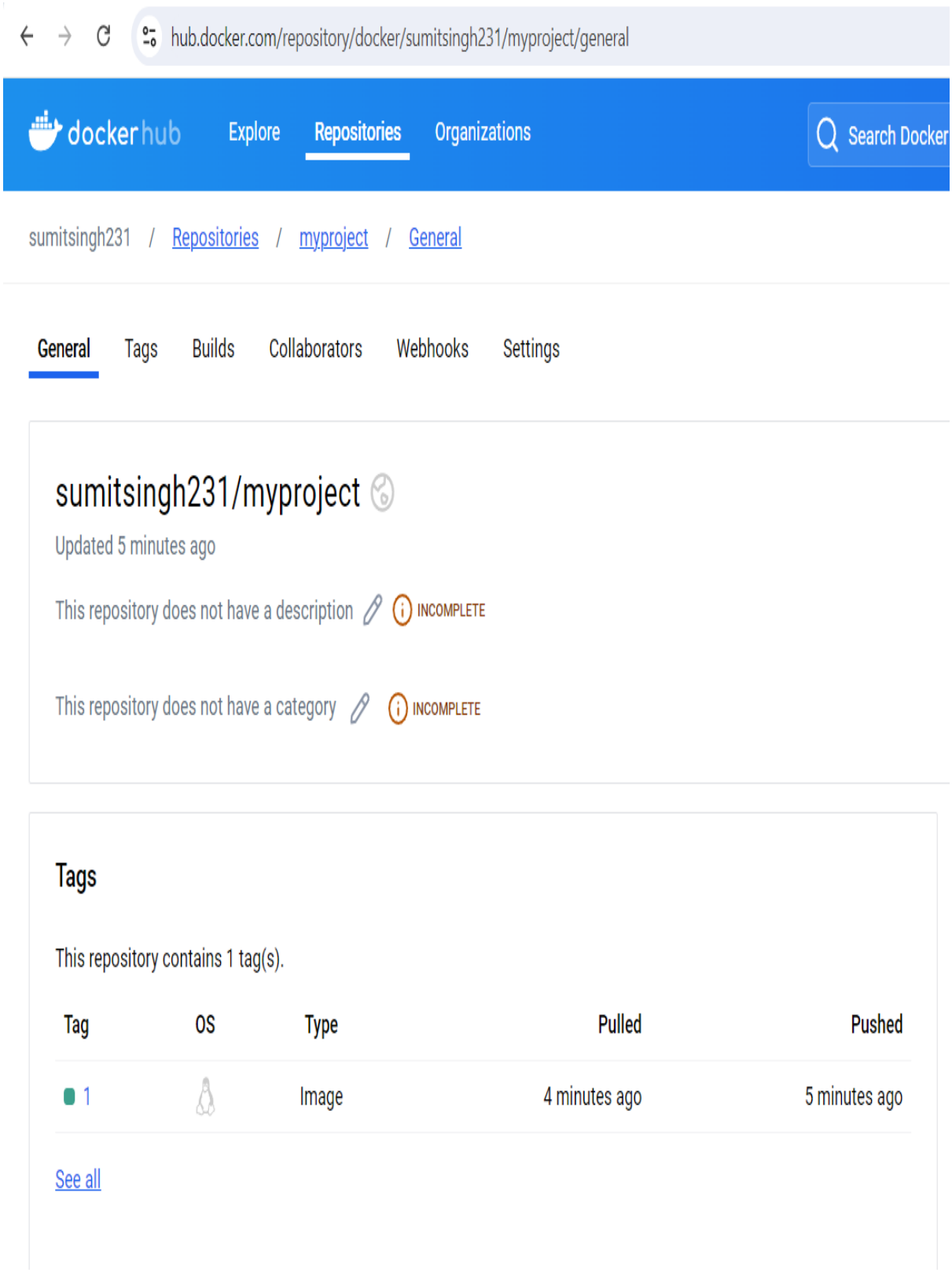


Step 29: Now go to docker hub account and verify image has been pushed

- **You can see image myproject:1 has been pushed successfully**
- **Last push 2 min ago**

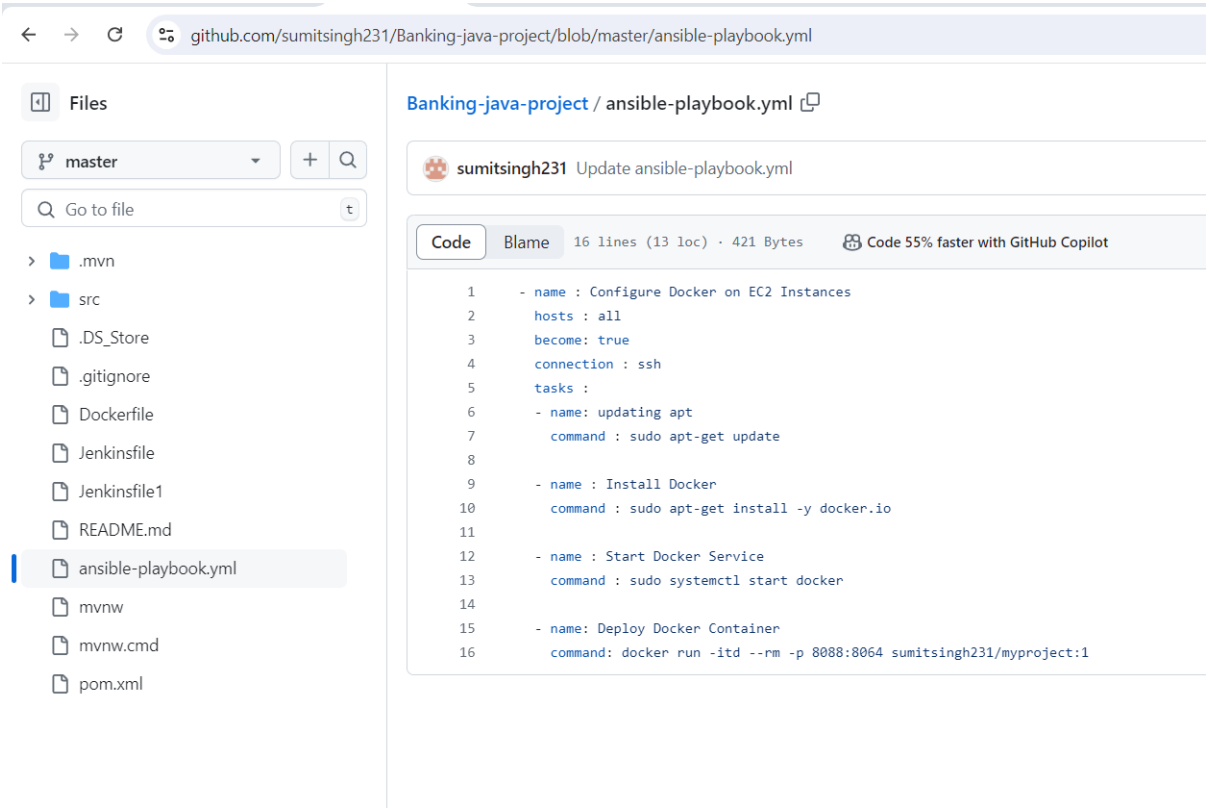


- **Click on myproject**
- **Here you can see the tag as 1**



- Now going to Github repository of Banking project and there
- Cllick on ansible-playbook.yml file
-
- Modify the content as below

$command: docker run -itd --rm -p 8088:8064 sumitsingh231/myproject:1



Step 30: Now go to master instance and execute below command for define host

$cd /etc/ansible/

```
root@ip-172-31-5-152:/home/ubuntu# service jenkins restart
root@ip-172-31-5-152:/home/ubuntu# cd /etc/ansible/
root@ip-172-31-5-152:/etc/ansible# ls
ansible.cfg  hosts  roles
root@ip-172-31-5-152:/etc/ansible#
```

i-0f5f05c6c2c216ee9 (Mater)

PublicIPs: 52.66.253.128 PrivateIPs: 172.31.5.152

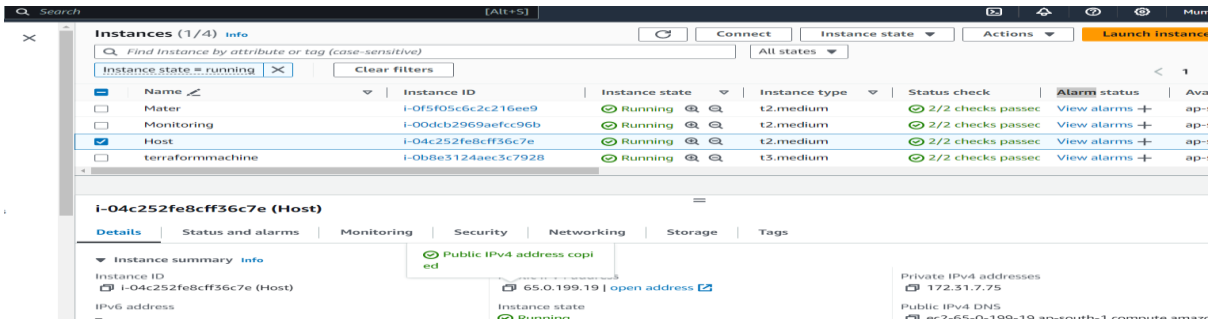
- **Now execute below command**

$vi hosts

- **Come to bottom and type**

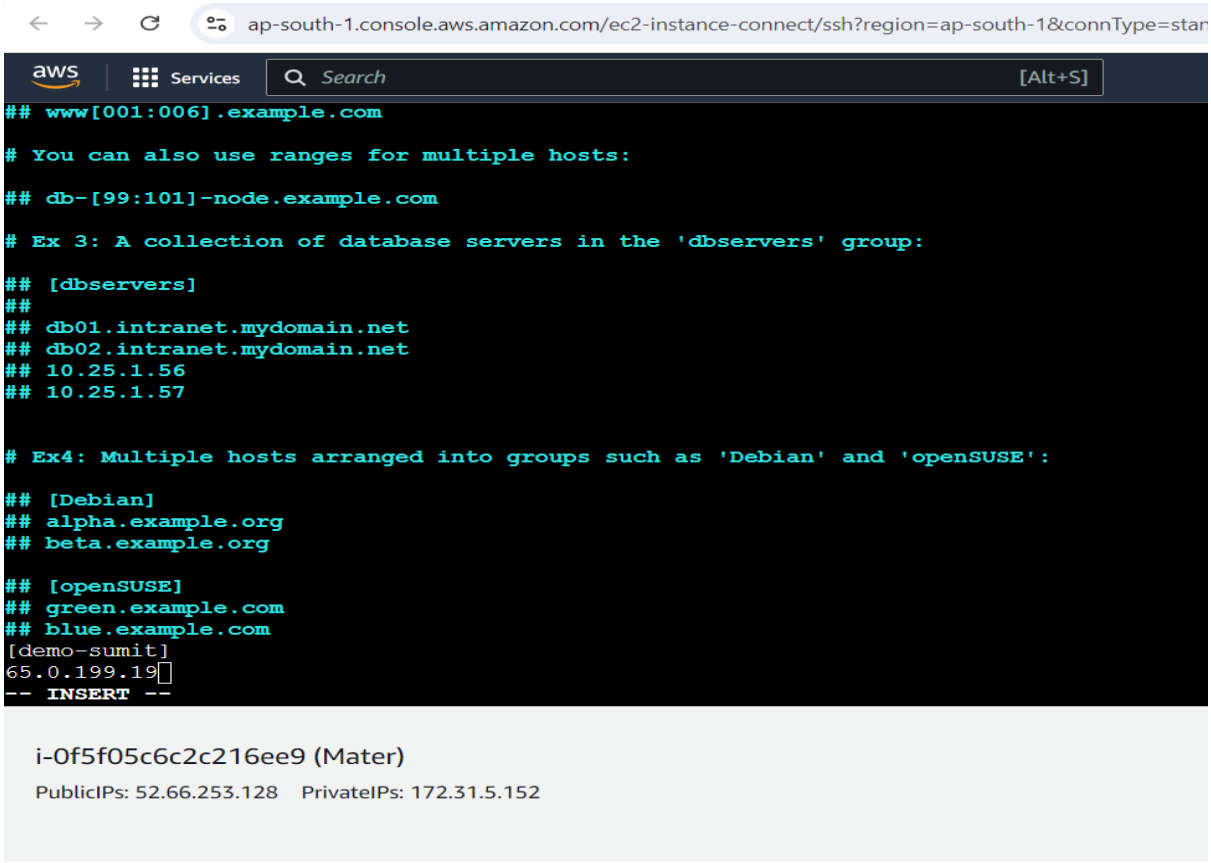
[demo-sumit]

Publicip of host server



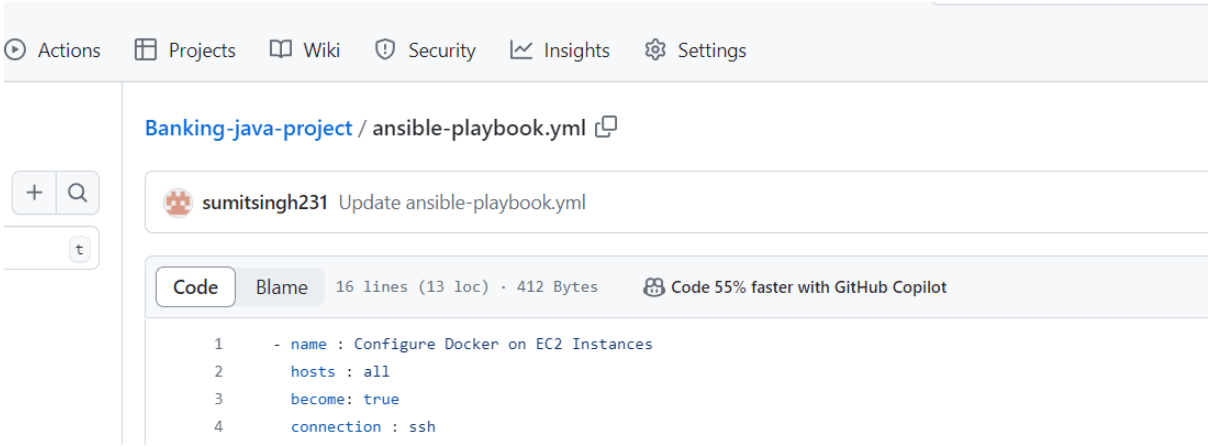
[demo-sumit]

65.0.199.19

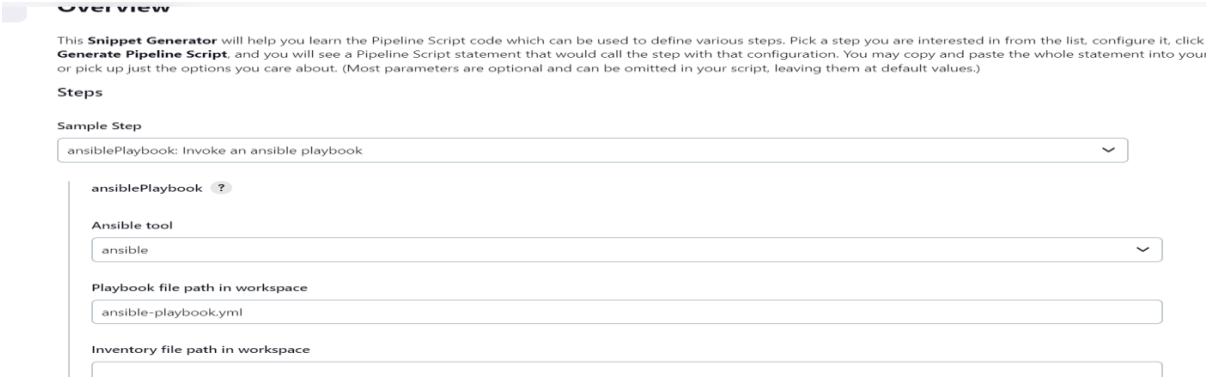


Step 31: Now come back to the pipeline

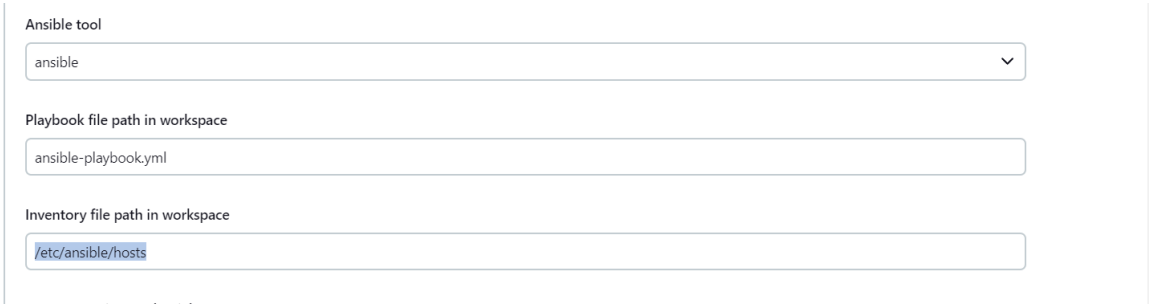
- click on pipeline syntax
- Select ansiblePlaybook:Invoke an ansible playbook
- Copy playbook file name



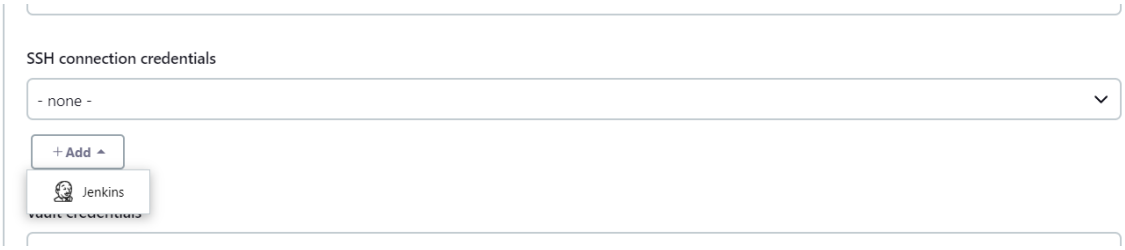
- Paste in playbook file path in workspace



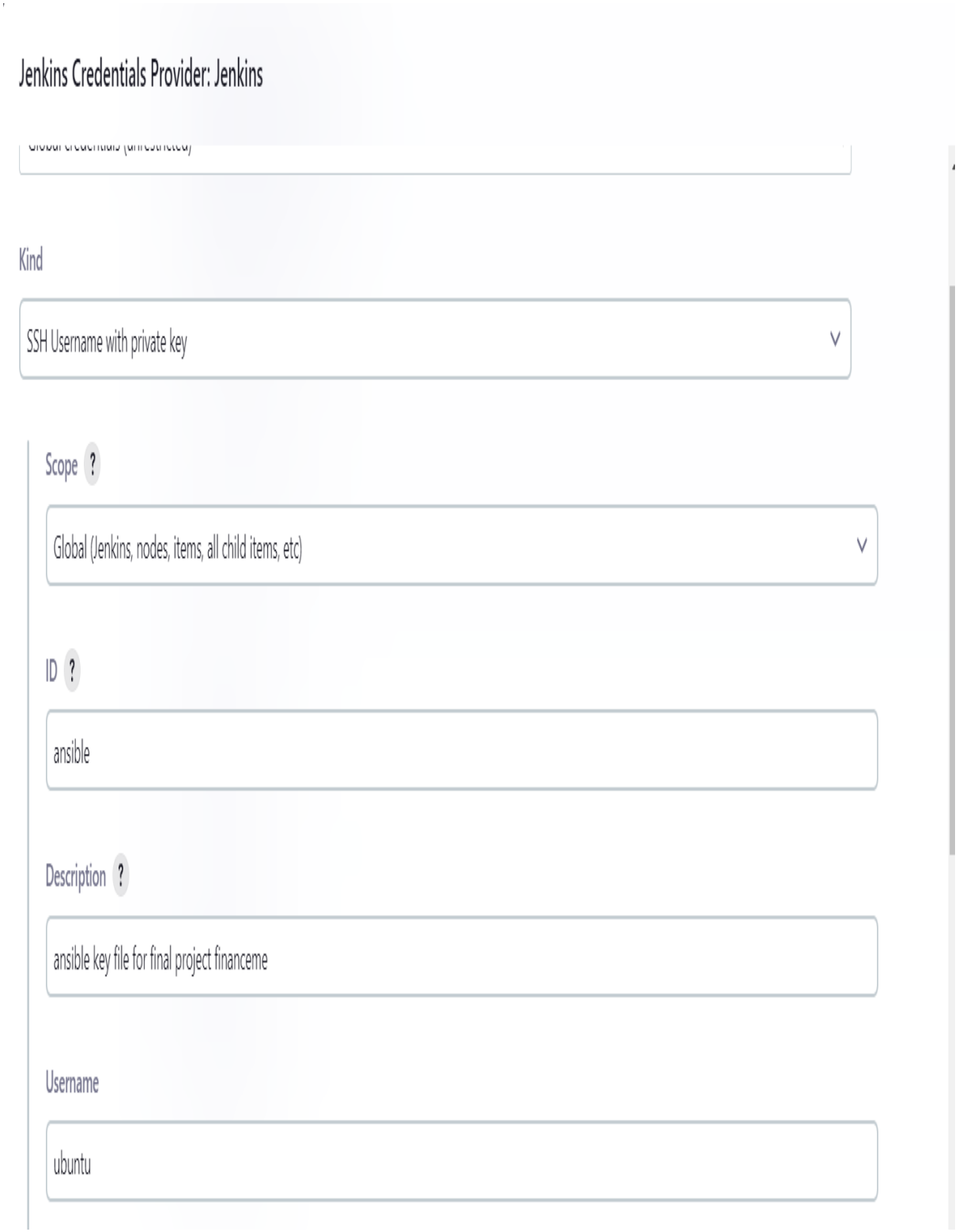
- **In Inventory file path type /etc/ansible/hosts**



- **In ssh connection credentials,click on add and select Jenkins**



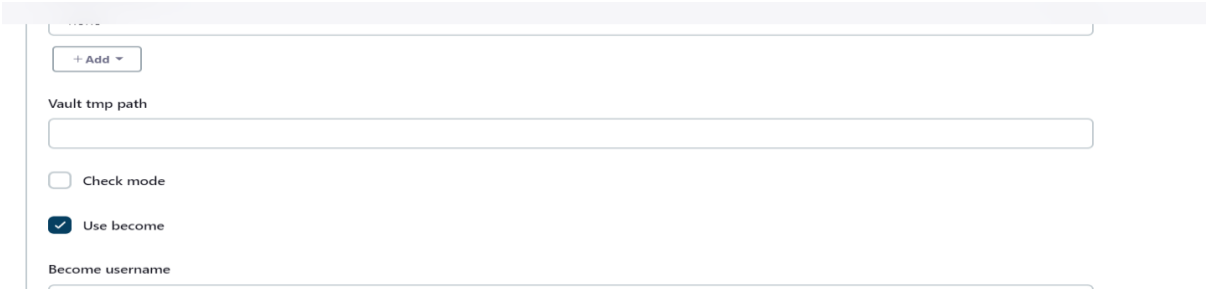
- **Select SSH connection with private key**
- **In ID type ansible**
- **In description type ansible key file for Final project (financeme)**
- **In username type ubuntu**



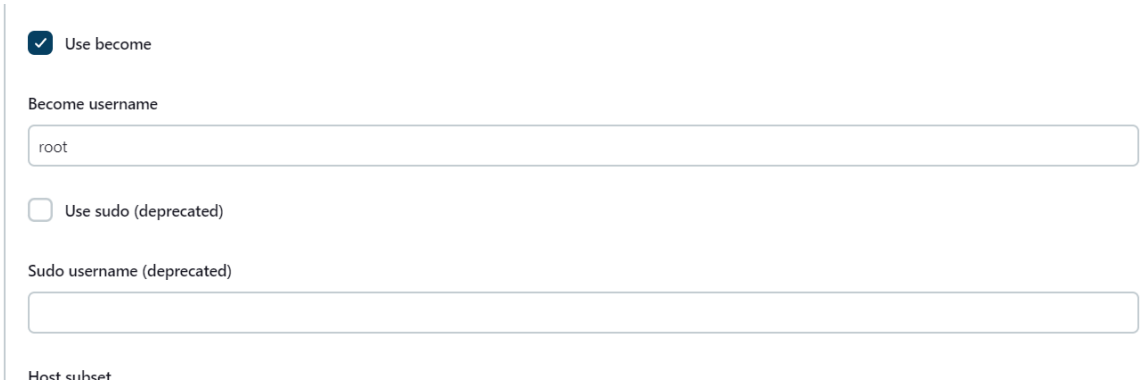
- **Click on Enter directly**
- **Now copy the key file of host I have used project1.pem key file so copying data of this key file and paste it into Key section then click on add**

- **Now in SSH connection credentials select which we have created**

- **Scroll down and checkout Use become**

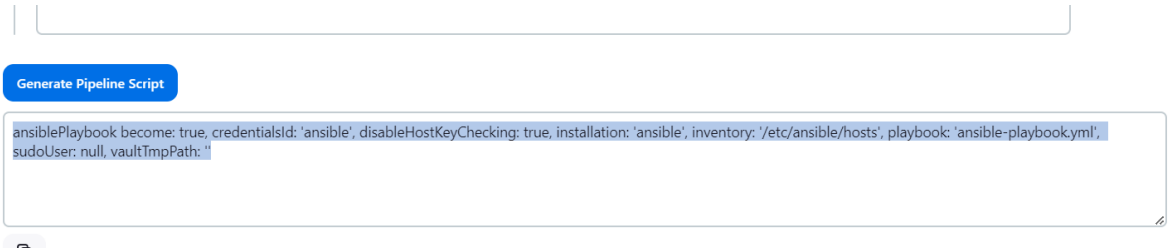


- **Become username type root and sudo username keep blank**

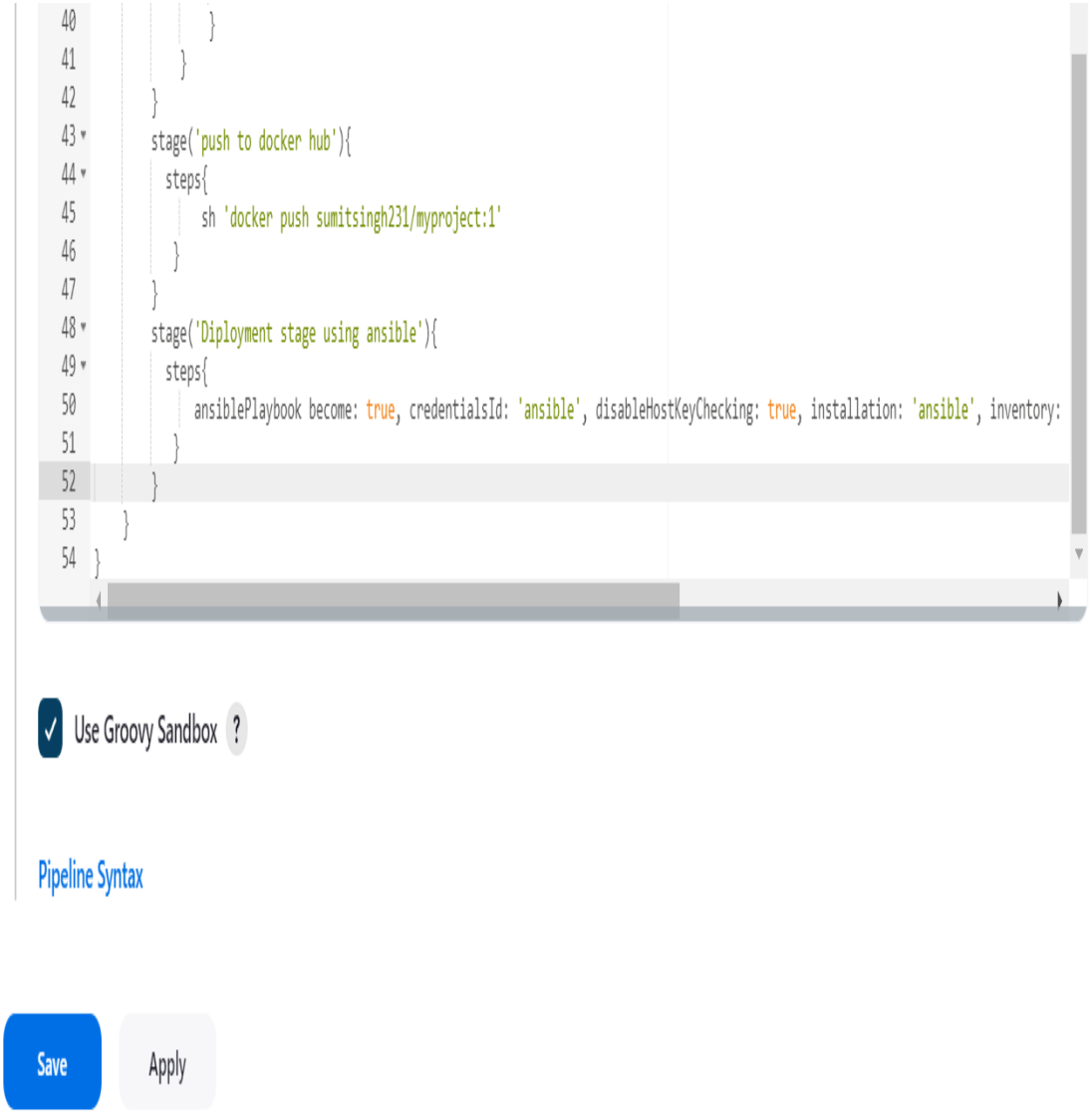


- **Checkout the Disable the host SSH key check,then click on Generate pipeline script**

- **Pipeline script is generated,copy it**



Step 32: Now go to the pipeline and paste the script into line number 50,then click on apply then save



Step 33: We have restarted Our Mater,Host ,and Monitor machine so public ip has changed

- **Master Machine-**
  Public ip: 13.201.75.139

```
ubuntu@ip-172-31-5-152:~$ ls
ansible.sh  jenkins.sh  node_exporter-1.8.2.linux-amd64  node_exporter-1.8.2.linux-amd64.tar.gz
ubuntu@ip-172-31-5-152:~$
```

i-0f5f05c6c2c216ee9 (Master)

PublicIPs: 13.201.75.139 PrivateIPs: 172.31.5.152

- **Host machine-**
  Public ip: 65.0.27.52

```
node_exporter-1.8.2.linux-amd64  node_exporter-1.8.2.linux-amd64.
root@ip-172-31-7-75:/home/ubuntu# docker ps -a
CONTAINER ID   IMAGE                          COMMAND
49c0a61fcf96   sumitsingh231/myproject:1   "java -jar /app.jar"
root@ip-172-31-7-75:/home/ubuntu#
```

i-04c252fe8cff36c7e (Host)

PublicIPs: 65.0.27.52 PrivateIPs: 172.31.7.75

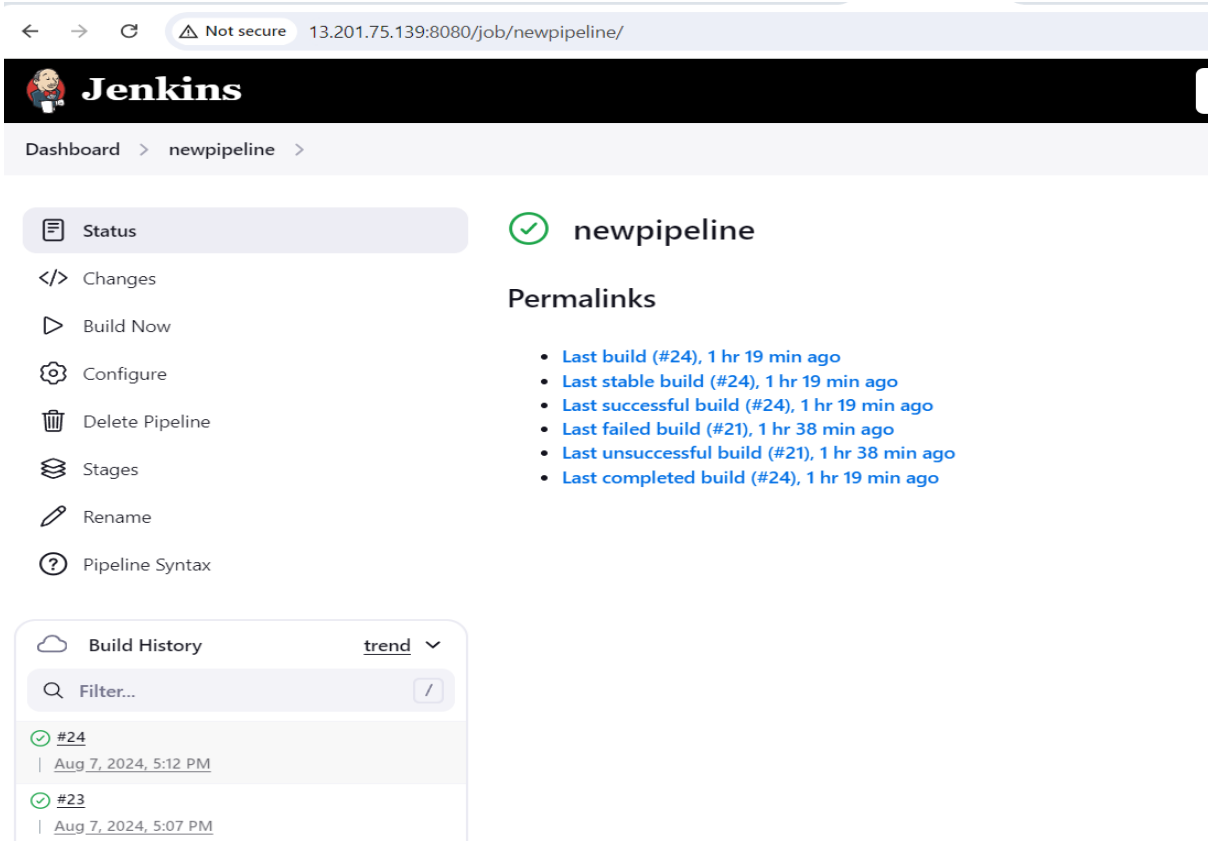
- **Monitoring Machine-**
  Public ip: 13.233.153.5

```
ts=2024-08-07T18:02:05.902Z caller=main.go:1415 level=in
ge=0s remote_storage=1.762µs web_handler=0s query_engine
5µs
ts=2024-08-07T18:02:05.902Z caller=main.go:1145 level=in
ts=2024-08-07T18:02:05.902Z caller=manager.go:164 level=
```

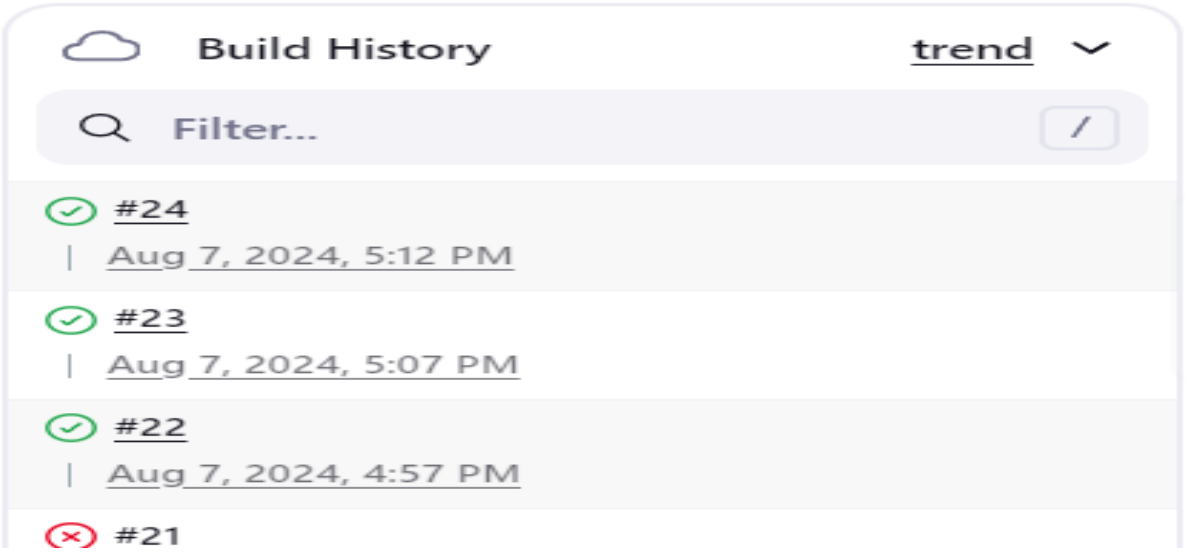
i-00dcb2969aefcc96b (Monitoring)

PublicIPs: 13.233.153.5 PrivateIPs: 172.31.15.199

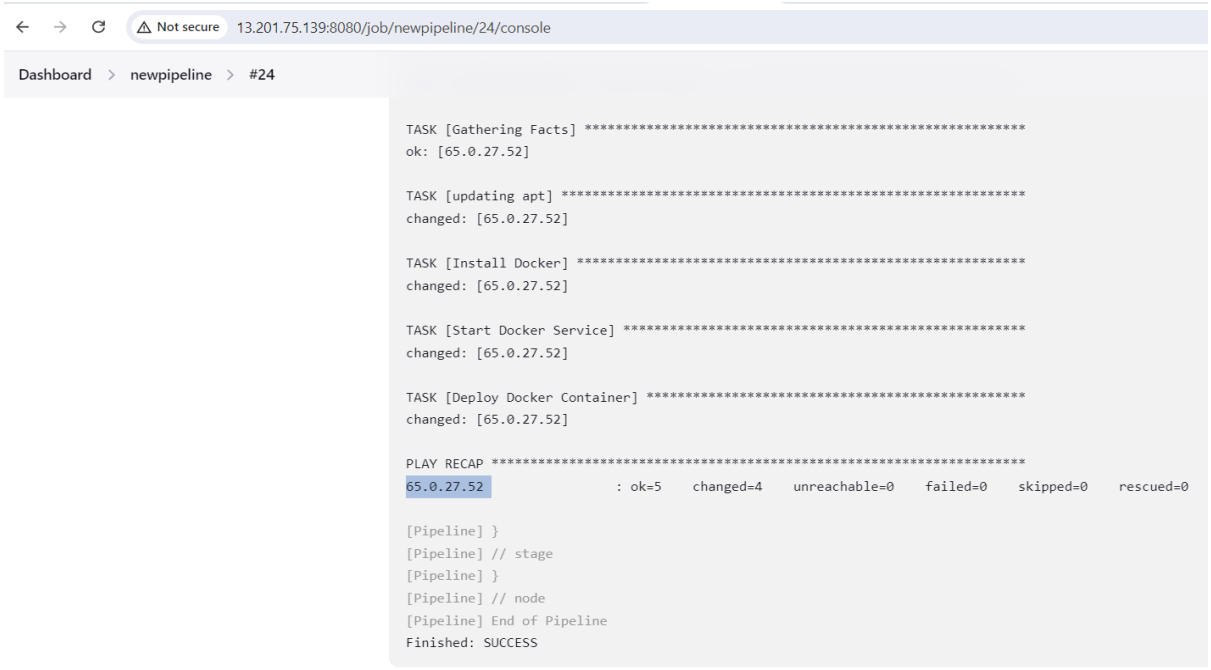
Step 34: Now again click on build now,you can see #24 Successfully build



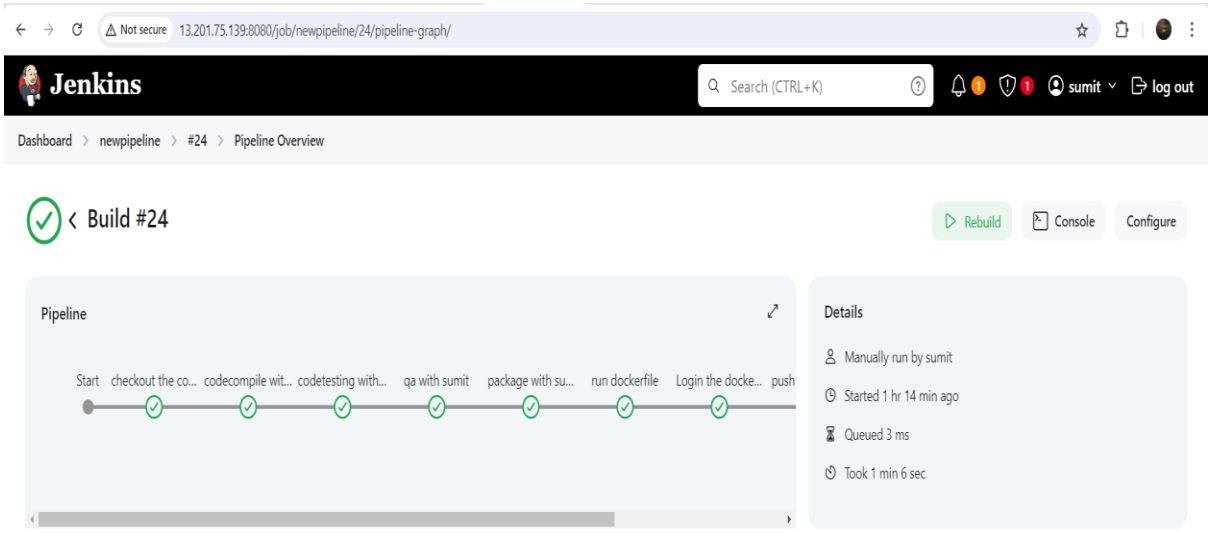
- **Build history-**



❖ **In console output build is successfully finished**

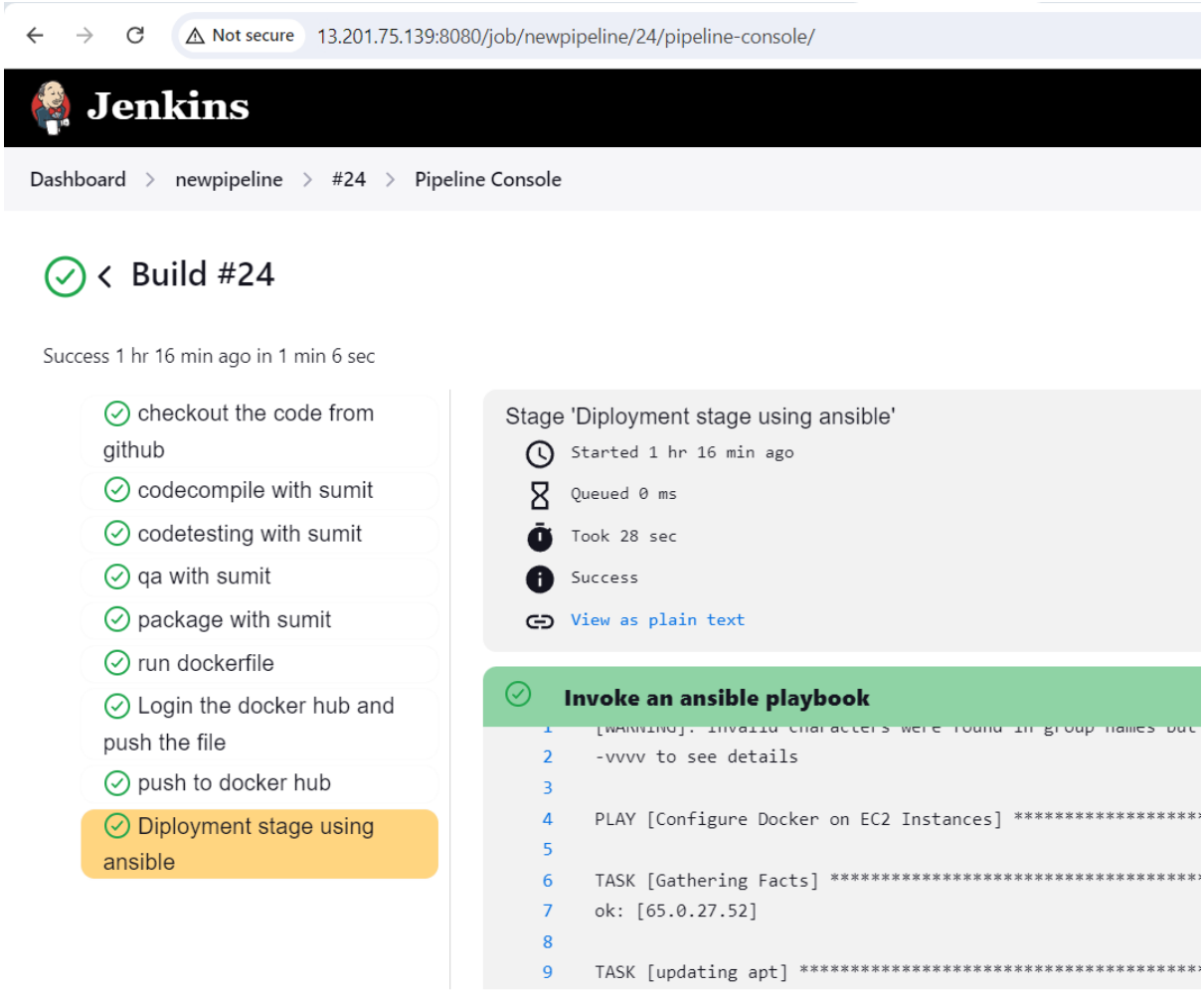


❖ You can also see in pipeline overview



- **Expand View-**

❖ Now you also can see the pipeline console-



Step 35: Now connect the host server ans execute below command in this

$docker ps

$docker images

```
root@ip-172-31-7-75:/home/ubuntu# docker ps
CONTAINER ID   IMAGE                     COMMAND             CREATED           STATUS           PORTS                                       NAMES
49c0a61fcf96   sumitsingh231/myproject:1   "java -jar /app.jar"   About an hour ago   Up About an hour   0.0.0.0:8088->8064/tcp, :::8088->8064/tcp   modest_bose
root@ip-172-31-7-75:/home/ubuntu# docker images
REPOSITORY              TAG       IMAGE ID       CREATED       SIZE
sumitsingh231/myproject   1         ce22249cf0db   27 hours ago   696MB
root@ip-172-31-7-75:/home/ubuntu#
```

i-04c252fe8cff36c7e (Host)

PublicIPs: 65.0.27.52 PrivateIPs: 172.31.7.75

- **cropped image-**

- **Now by using below ip (Host Machine public ip)and port 8088 we can access the project**

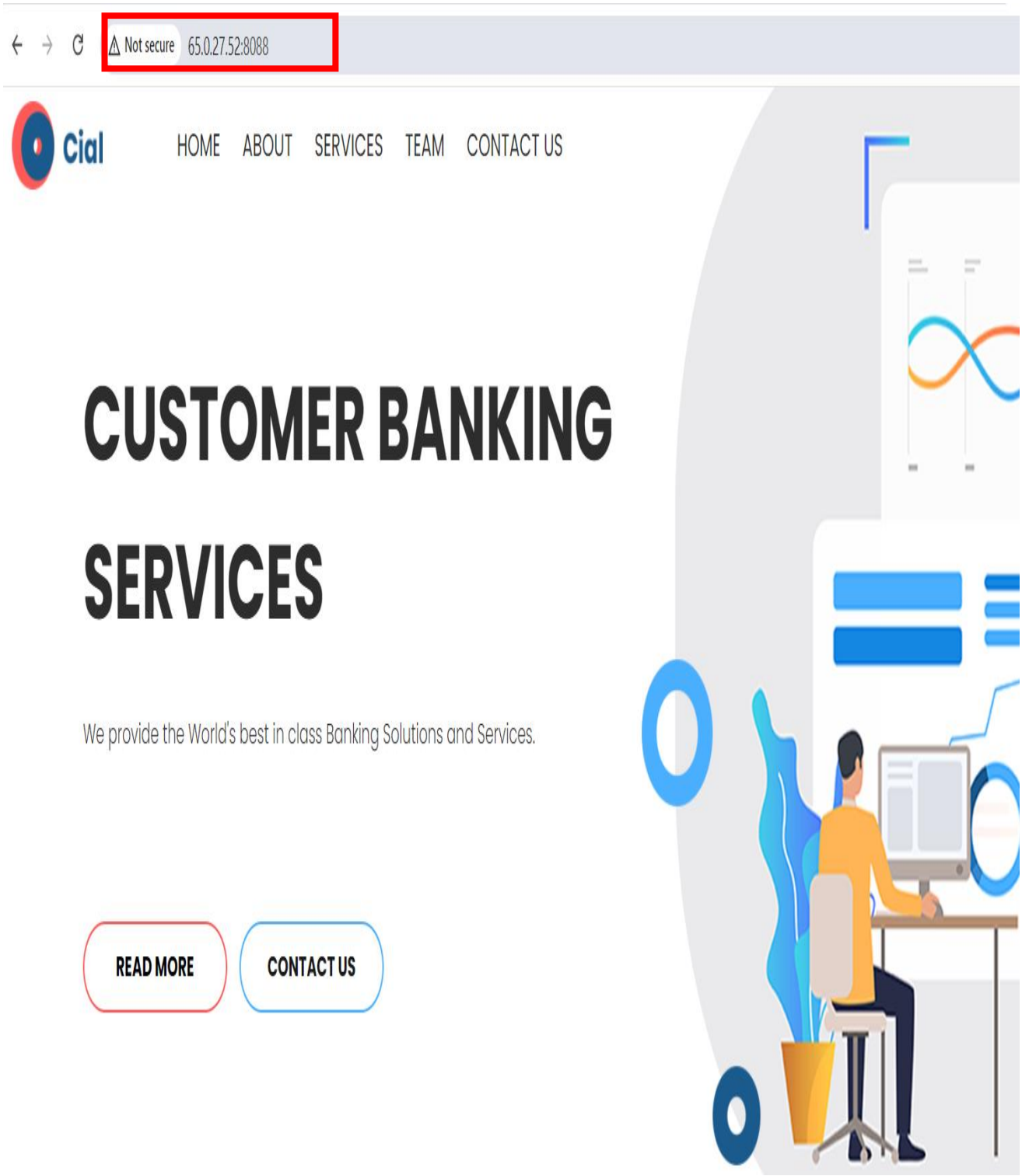
docker build -t sumitsingh231/myproject:1 .

```
PLAY RECAP *********************************************************************
65.0.27.52                 : ok=5    changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

docker build -t sumitsingh231/myproject:1 .

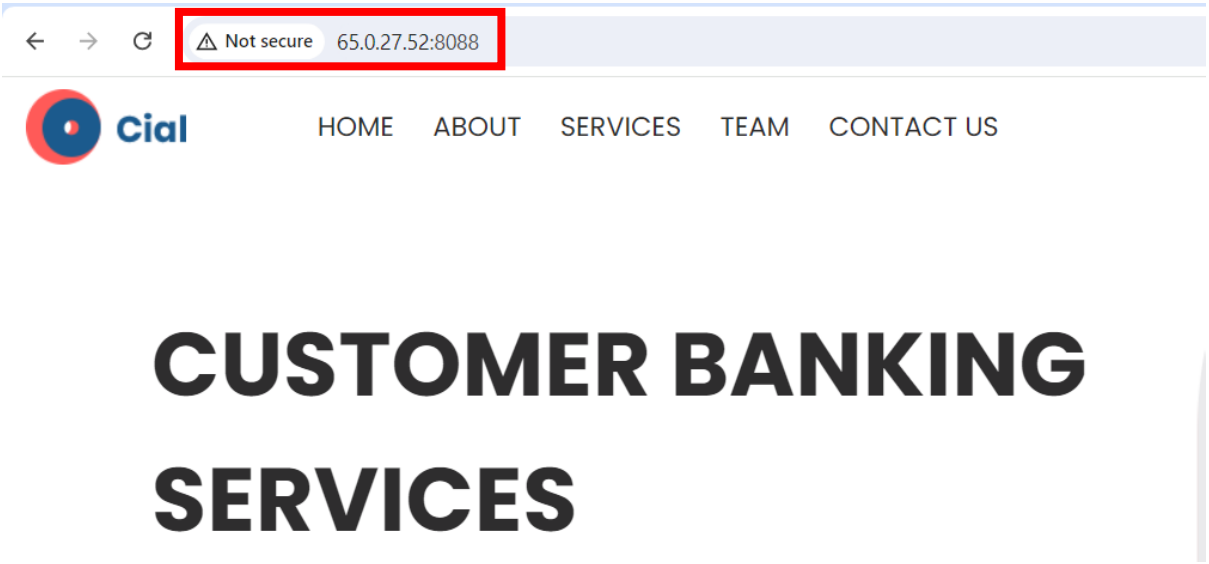
Step 36: Now browse public ip of host machine with port 8088

http:// 65.0.27.52:8088



You can see we can access our banking project(financeme)

Cropped image-



- ❖ **Hence we accessed it on browser by using Host machine ip address**

Step 37: Now installing Prometheus in Monitoring machine

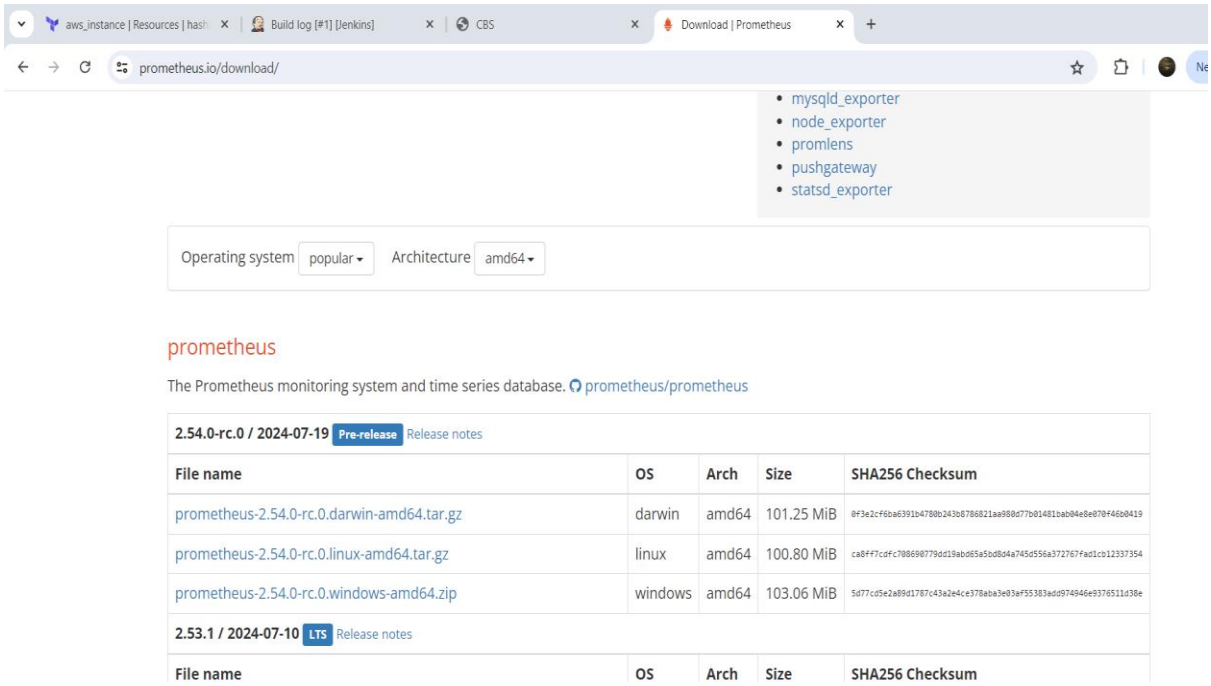
- **Connect the instance and execute below command**

$sudo su

- **Now create a file(prom.sh) by execute below command**

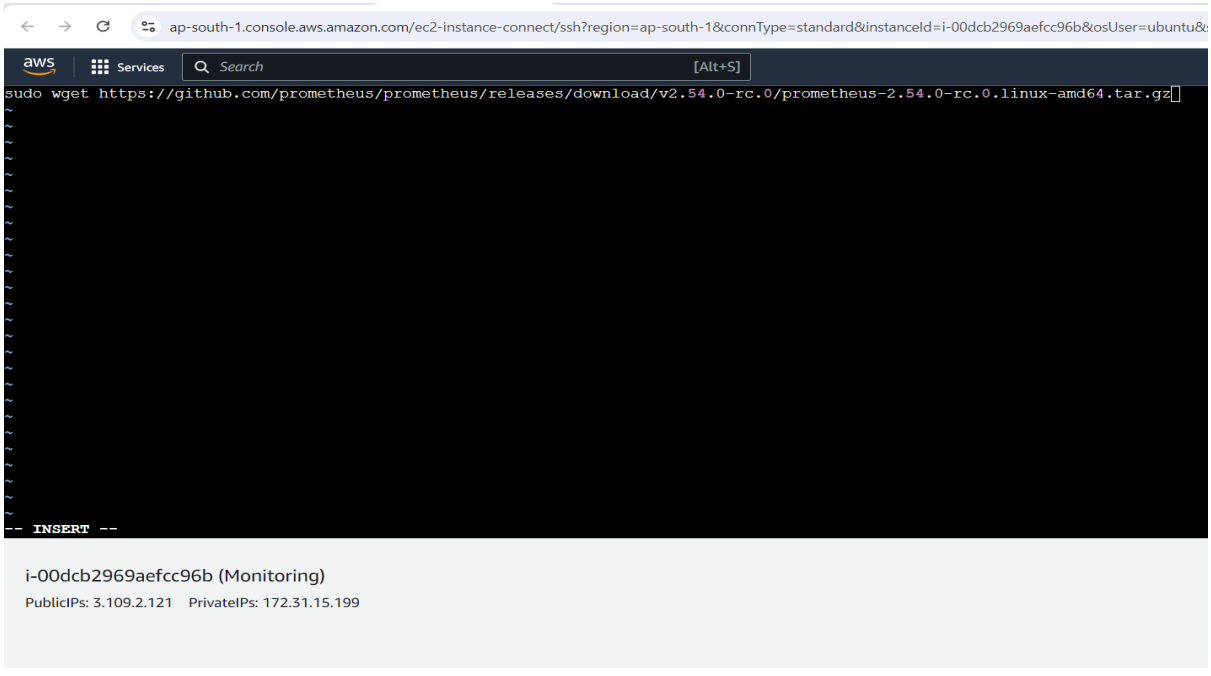
$vi prom.sh

- **Now browse install Prometheus on ubuntu on browser**
- **Open official website**
- **Click on download**



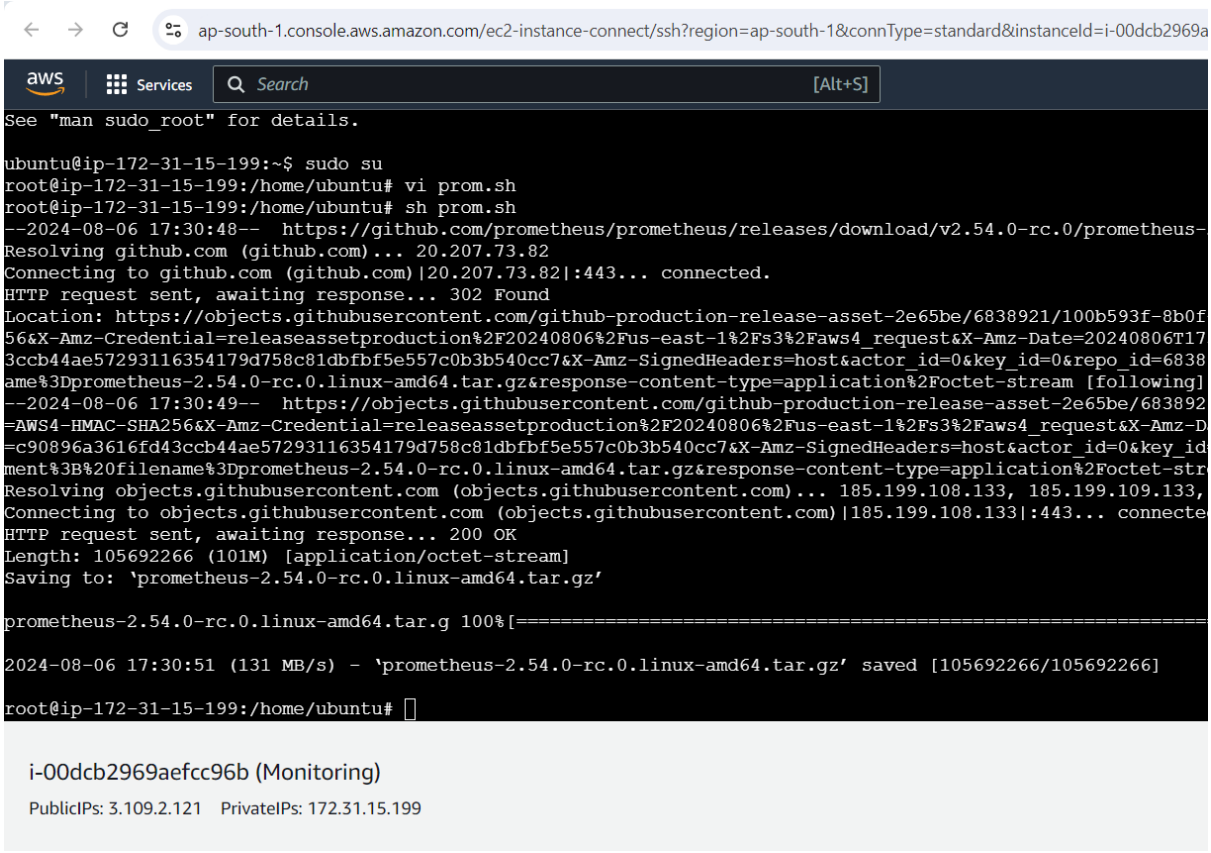
- **Copy link address and paste it into prom.sh file with sudo wget**

sudo wget https://github.com/prometheus/prometheus/releases/download/v2.54.0-rc.0/prometheus-2.54.0-rc.0.linux-amd64.tar.gz



- **Press esc--->:wq (For save and exit)**
- **Now execute below command to run prom.sh file**

$sh prom.sh



- **Now execute below command for unzip the file**

$ tar -xvzf prometheus-2.54.0-rc.0.linux-amd64.tar.gz

- **Now go inside to the folder prometheus-2.54.0-rc.0.linux-amd64 by execute below command**

$cd prometheus-2.54.0-rc.0.linux-amd64

```
root@ip-172-31-15-199:/home/ubuntu# ls
prom.sh  prometheus-2.54.0-rc.0.linux-amd64  prometheus-2.54.0-rc.0.linux-amd64.tar.gz
root@ip-172-31-15-199:/home/ubuntu# cd prometheus-2.54.0-rc.0.linux-amd64
root@ip-172-31-15-199:/home/ubuntu/prometheus-2.54.0-rc.0.linux-amd64# ls
LICENSE  NOTICE  console_libraries  consoles  prometheus  prometheus.yml  promtool
root@ip-172-31-15-199:/home/ubuntu/prometheus-2.54.0-rc.0.linux-amd64#
```

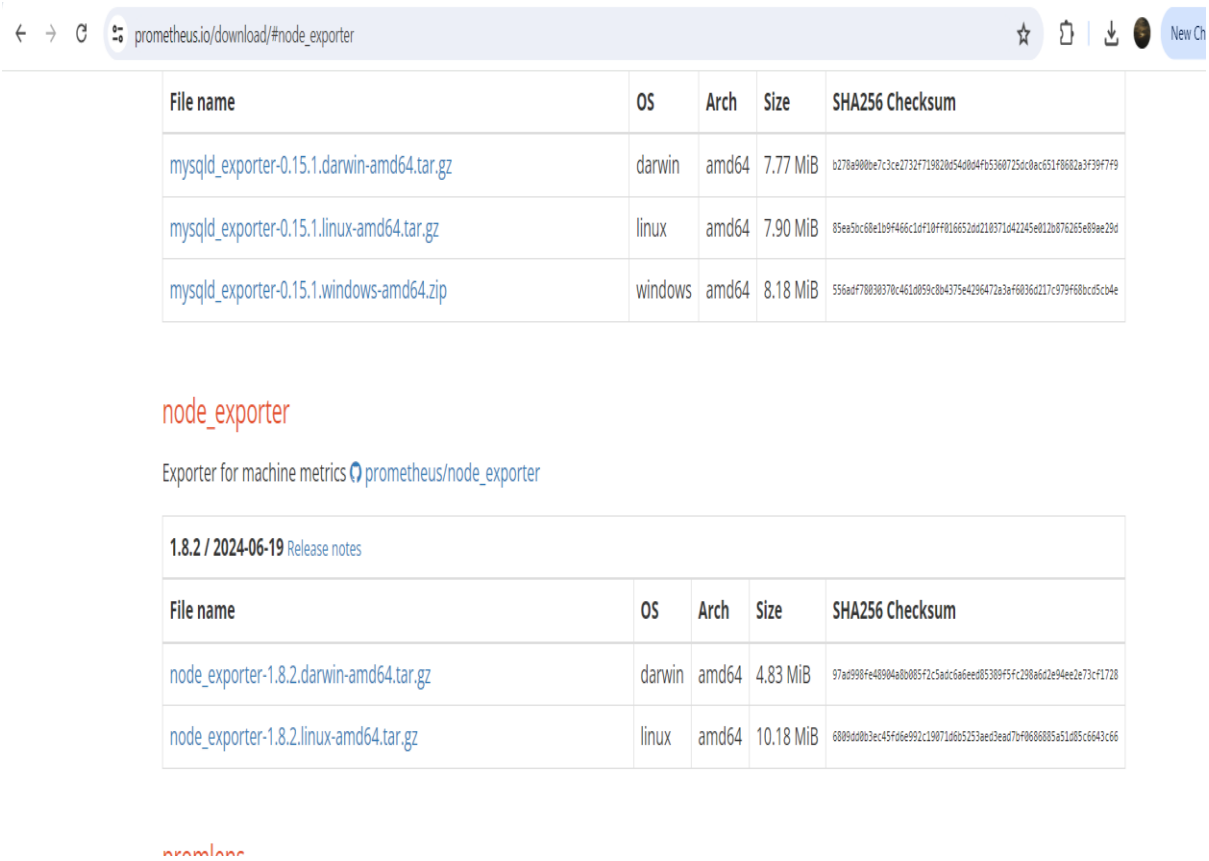
i-00dcb2969aefcc96b (Monitoring)

PublicIPs: 3.109.2.121 PrivateIPs: 172.31.15.199

Step 38: Now download node exporter in Host machine And Master Machine

- **For this go to browser and browse official document**

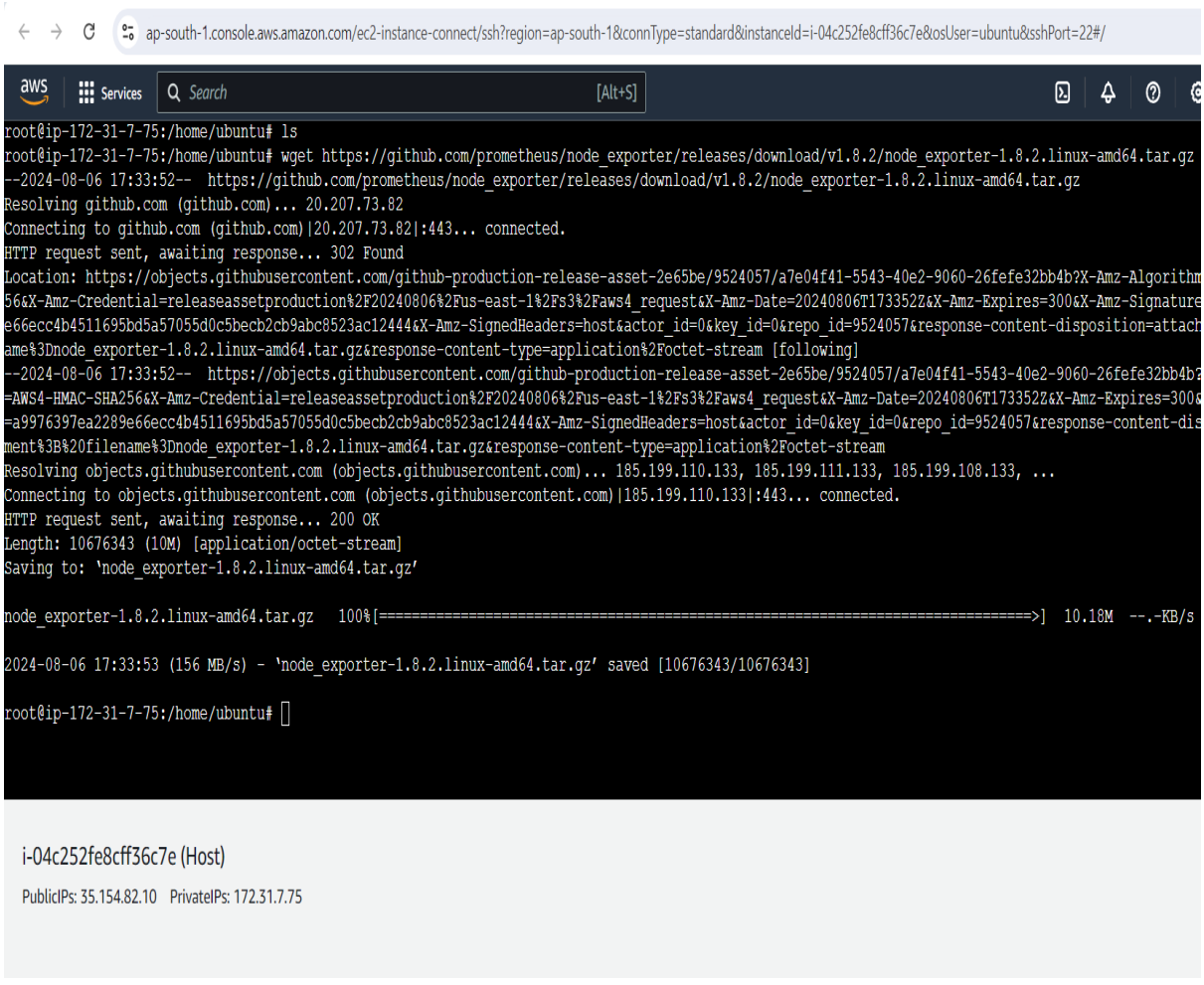
https://prometheus.io/download/#node_exporter



- **Now copy the link**

https://github.com/prometheus/node_exporter/releases/download/v1.8.2/node_exporter-1.8.2.linux-amd64.tar.gz

- **Now go to Host machine and execute below command**
  $wget copied link
  $wget https://github.com/prometheus/node_exporter/releases/download/v1.8.2/node_exporter-1.8.2.linux-amd64.tar.gz



- **Now execute below command to unzip the file**

$ tar xvfz node_exporter-1.8.2.linux-amd64.tar.gz

```
root@ip-172-31-7-75:/home/ubuntu# ls
node_exporter-1.8.2.linux-amd64.tar.gz
root@ip-172-31-7-75:/home/ubuntu# tar xvfz node_exporter-1.8.2.linux-amd64.tar.gz
node_exporter-1.8.2.linux-amd64/
node_exporter-1.8.2.linux-amd64/NOTICE
node_exporter-1.8.2.linux-amd64/node_exporter
node_exporter-1.8.2.linux-amd64/LICENSE
root@ip-172-31-7-75:/home/ubuntu# ls
node_exporter-1.8.2.linux-amd64  node_exporter-1.8.2.linux-amd64.tar.gz
root@ip-172-31-7-75:/home/ubuntu#
```

i-04c252fe8cff36c7e (Host)

PublicIPs: 35.154.82.10 PrivateIPs: 172.31.7.75

- **Now go to the folder**

$cd node_exporter-1.8.2.linux-amd64

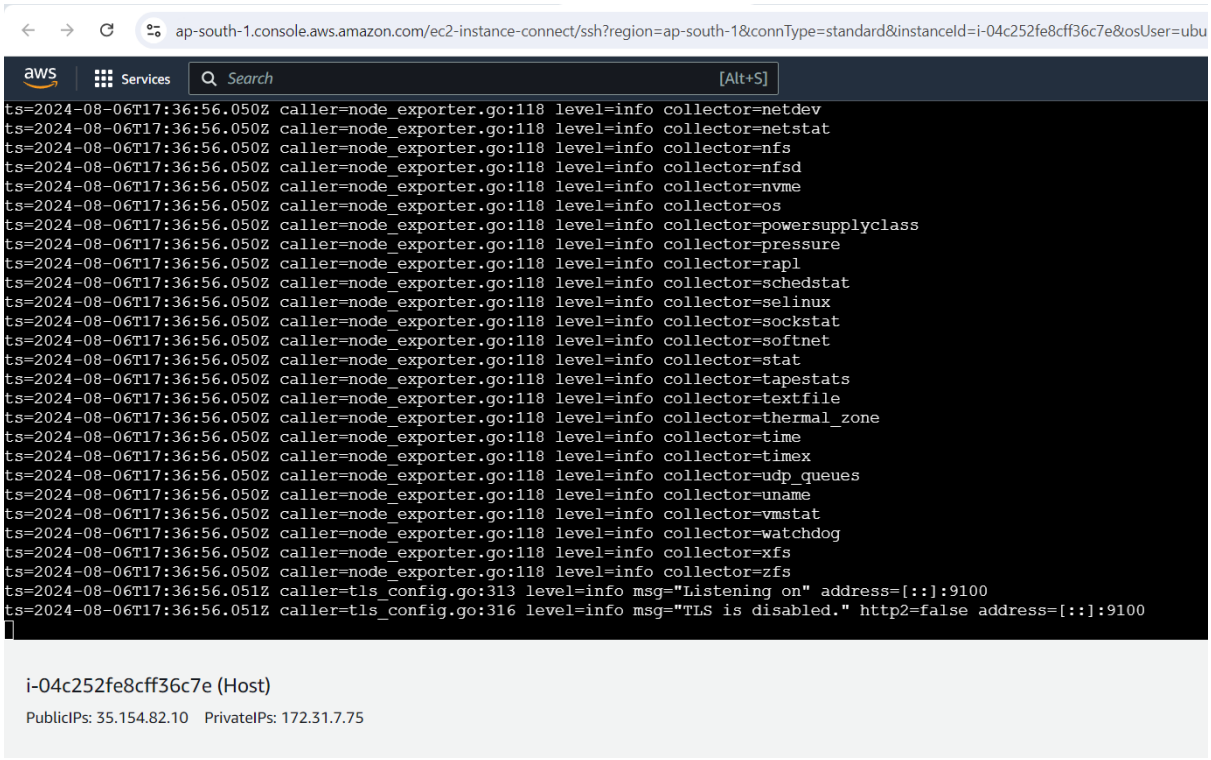
```
root@ip-172-31-7-75:/home/ubuntu# ls
node_exporter-1.8.2.linux-amd64  node_exporter-1.8.2.linux-amd64.tar.gz
root@ip-172-31-7-75:/home/ubuntu# cd node_exporter-1.8.2.linux-amd64
root@ip-172-31-7-75:/home/ubuntu/node_exporter-1.8.2.linux-amd64# ls
LICENSE  NOTICE  node_exporter
root@ip-172-31-7-75:/home/ubuntu/node_exporter-1.8.2.linux-amd64#
```

i-04c252fe8cff36c7e (Host)

PublicIPs: 35.154.82.10 PrivateIPs: 172.31.7.75

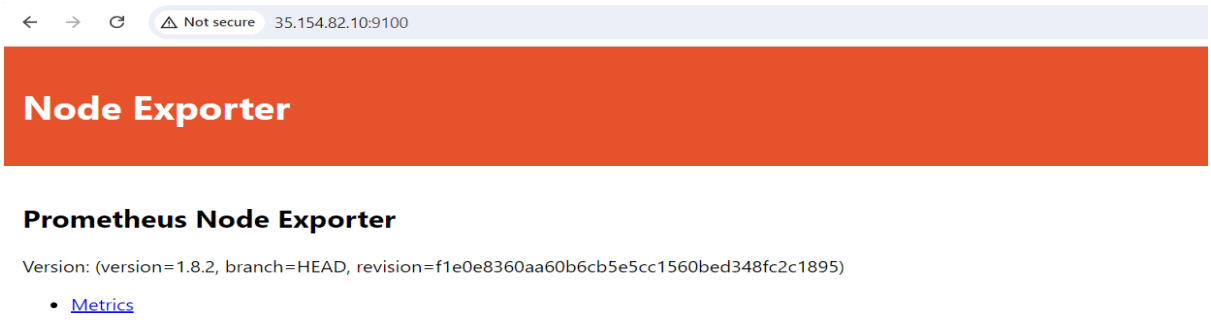
- **Now execute below command**

$ ./node_exporter



- **For verify go to browser and browse**

Publicip of Host machine instance:9100



- **Public ip of instance:9100/metrics**

Not secure 35.154.82.10:9100/metrics

```
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 0
go_gc_duration_seconds{quantile="0.25"} 0
go_gc_duration_seconds{quantile="0.5"} 0
go_gc_duration_seconds{quantile="0.75"} 0
go_gc_duration_seconds{quantile="1"} 0
go_gc_duration_seconds_sum 0
go_gc_duration_seconds_count 0
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 7
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.22.5"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 834440
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 834440
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.448924e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 730
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 1.80124e+06
# HELP go_memstats_heap_alloc_bytes Number of heap bytes allocated and still in use.
# TYPE go_memstats_heap_alloc_bytes gauge
go_memstats_heap_alloc_bytes 834440
# HELP go_memstats_heap_idle_bytes Number of heap bytes waiting to be used.
# TYPE go_memstats_heap_idle_bytes gauge
go_memstats_heap_idle_bytes 1.589248e+06
# HELP go_memstats_heap_inuse_bytes Number of heap bytes that are in use.
# TYPE go_memstats_heap_inuse_bytes gauge
go_memstats_heap_inuse_bytes 2.277376e+06
# HELP go_memstats_heap_objects Number of allocated objects.
# TYPE go_memstats_heap_objects gauge
go_memstats_heap_objects 8112
# HELP go_memstats_heap_released_bytes Number of heap bytes released to OS.
# TYPE go_memstats_heap_released_bytes gauge
go_memstats_heap_released_bytes 1.589248e+06
```

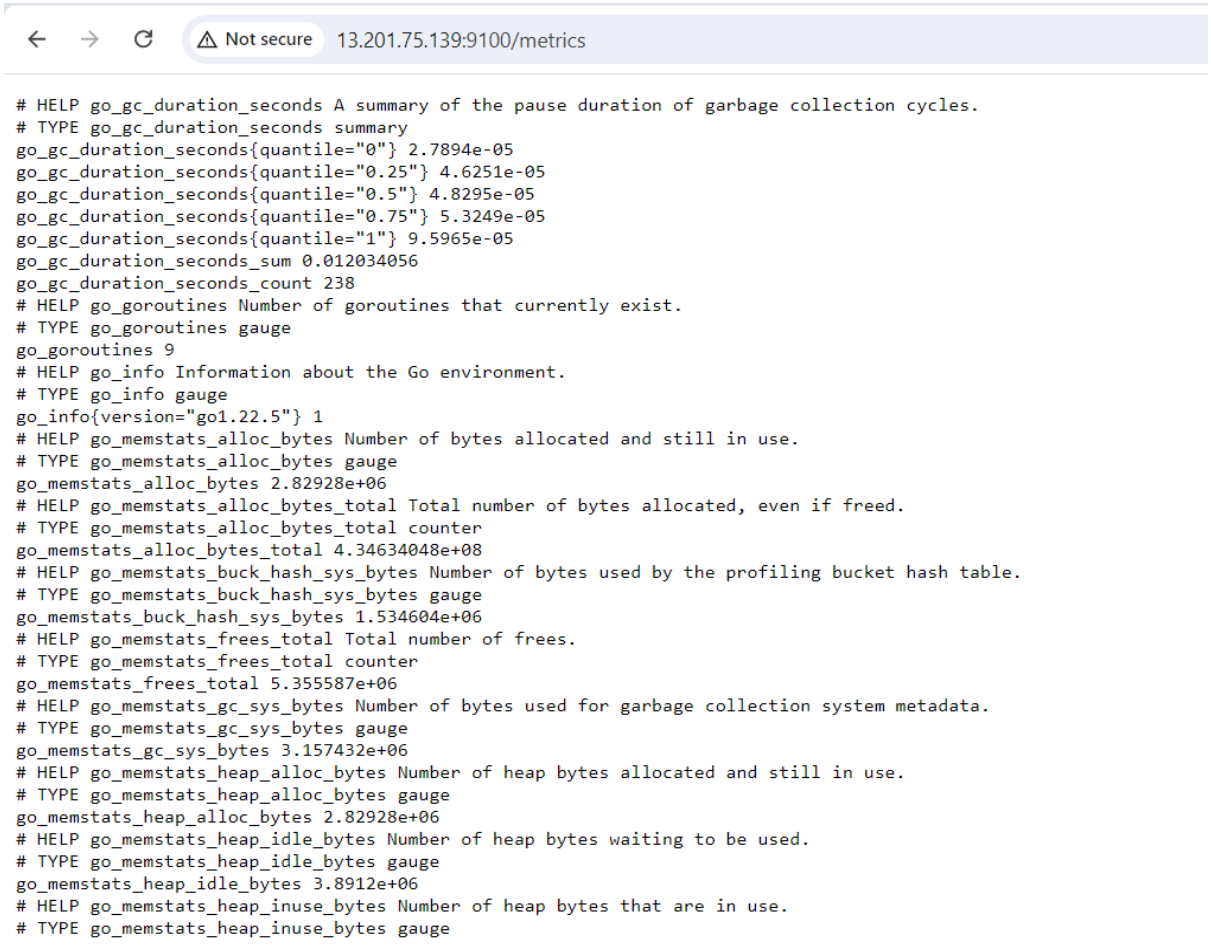
- **It shows all the metrics**
- **You can see we have configured Node Exporter in Host Machine-**

```
root@ip-172-31-7-75:/home/ubuntu# ls
node_exporter-1.8.2.linux-amd64  node_exporter-1.8.2.linux-amd64.tar.gz
root@ip-172-31-7-75:/home/ubuntu#
```

i-04c252fe8cff36c7e (Host)

- **By using same steps we installed Node Exporter in master machine also**

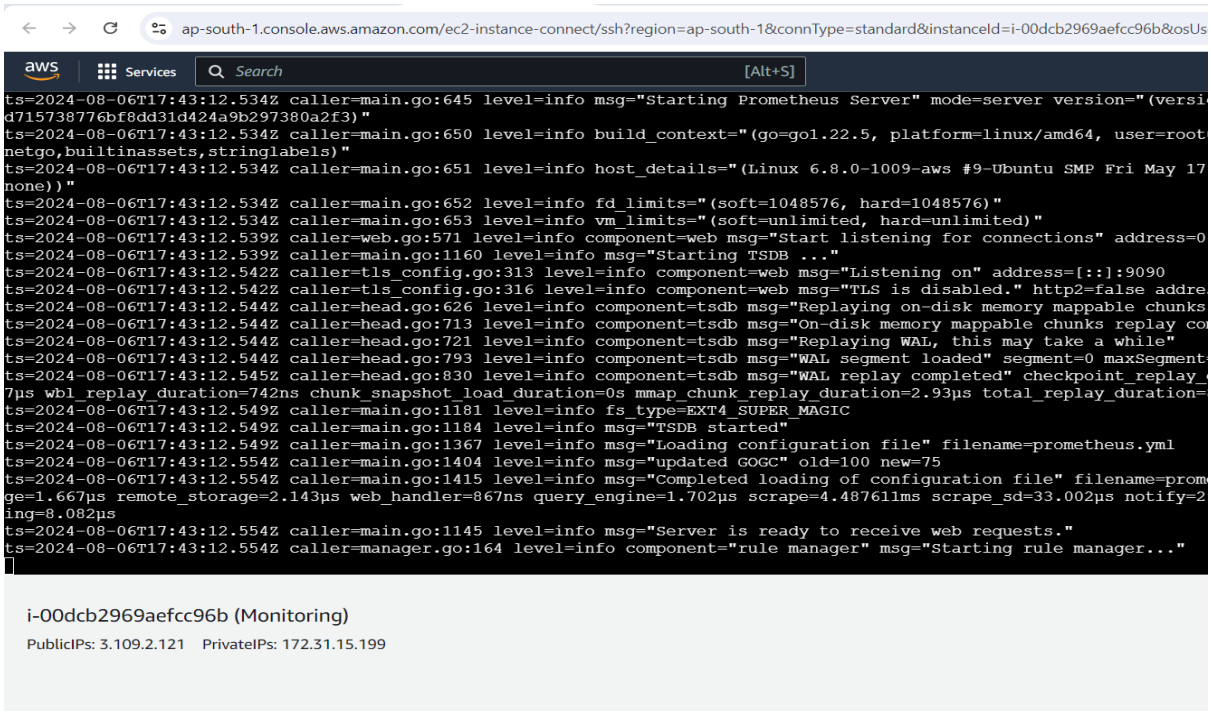
Publicip of Master machine instance:9100



- **You can see we have Installed Node Exporter in Master Machine**

Step 39: Now go back to the monitoring server and execute below command

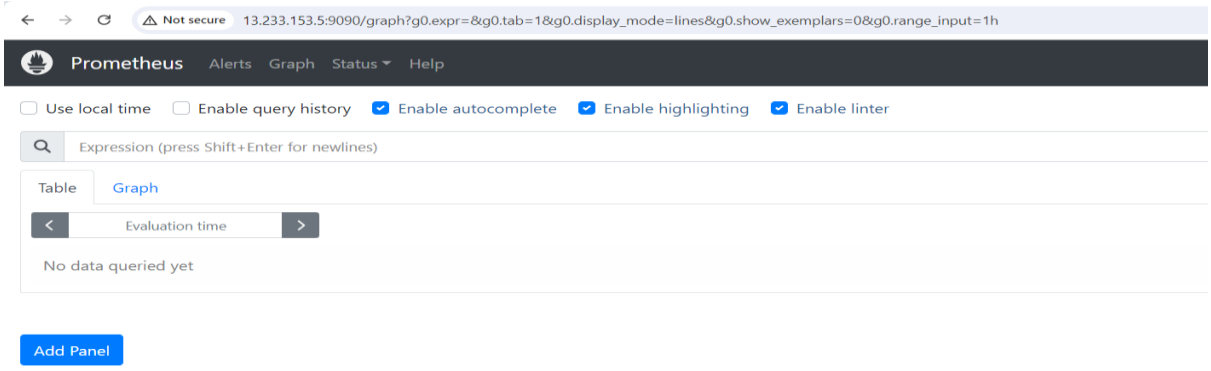
$ ./ Prometheus



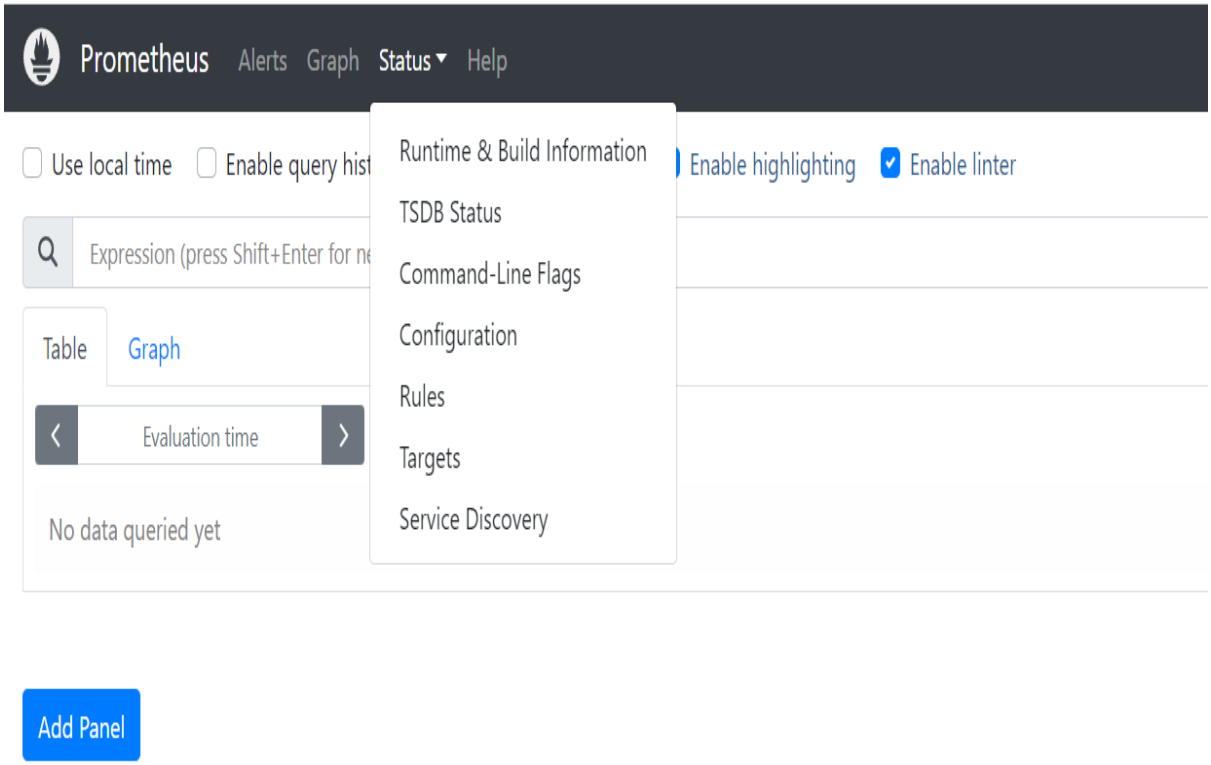
Step 40: Now copy the same public ip of monitoring instance with Prometheus default port 9090,search on browser

Public ip:9090 ----> it shows Prometheus

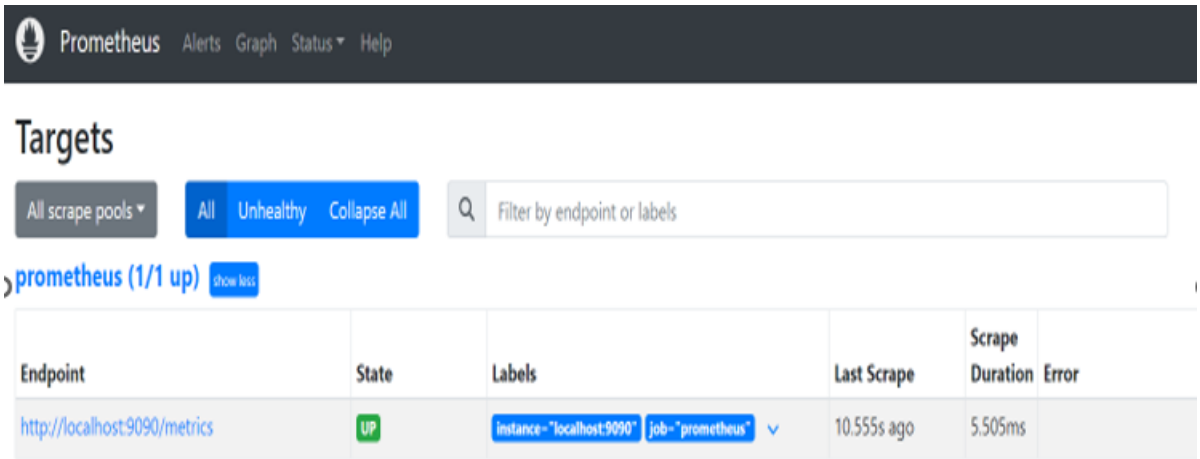
13.233.153.5:9090



- **Now click on status then targets**



- **We can see the state is up**



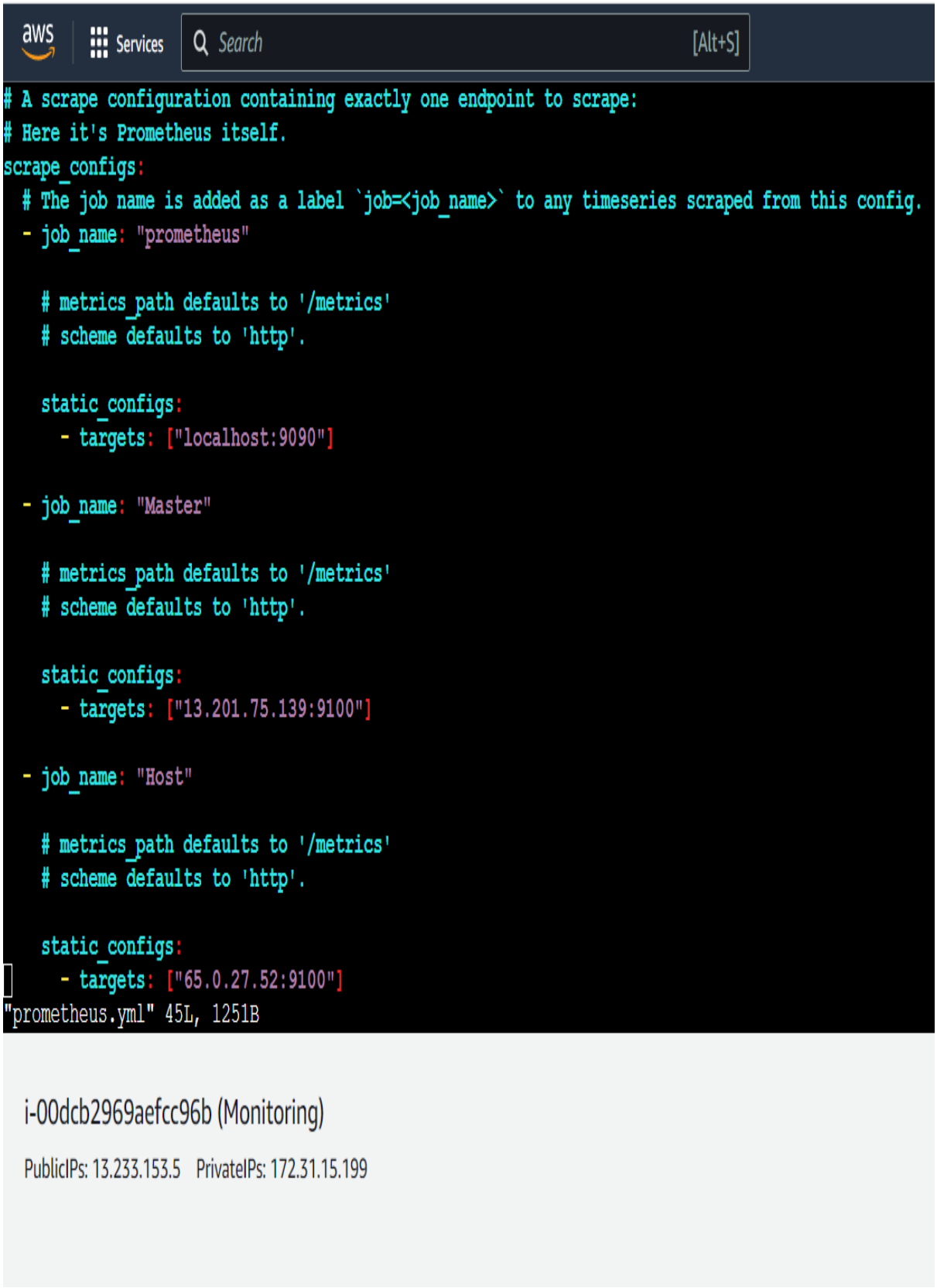
- **Now go to the prometheus.yml file be execute below command**

$vi Prometheus.yml

- Now define the target in this file,for this we edited the file
- Add below code at last,

```
-  - job_name: "Master"

  # metrics_path defaults to '/metrics'

  # scheme defaults to 'http'.

  static_configs:

   - targets: ["13.201.75.139:9100"]

- job_name: "Host"

  # metrics_path defaults to '/metrics'

  # scheme defaults to 'http'.

  static_configs:

   - targets: ["65.0.27.52:9100"]
```

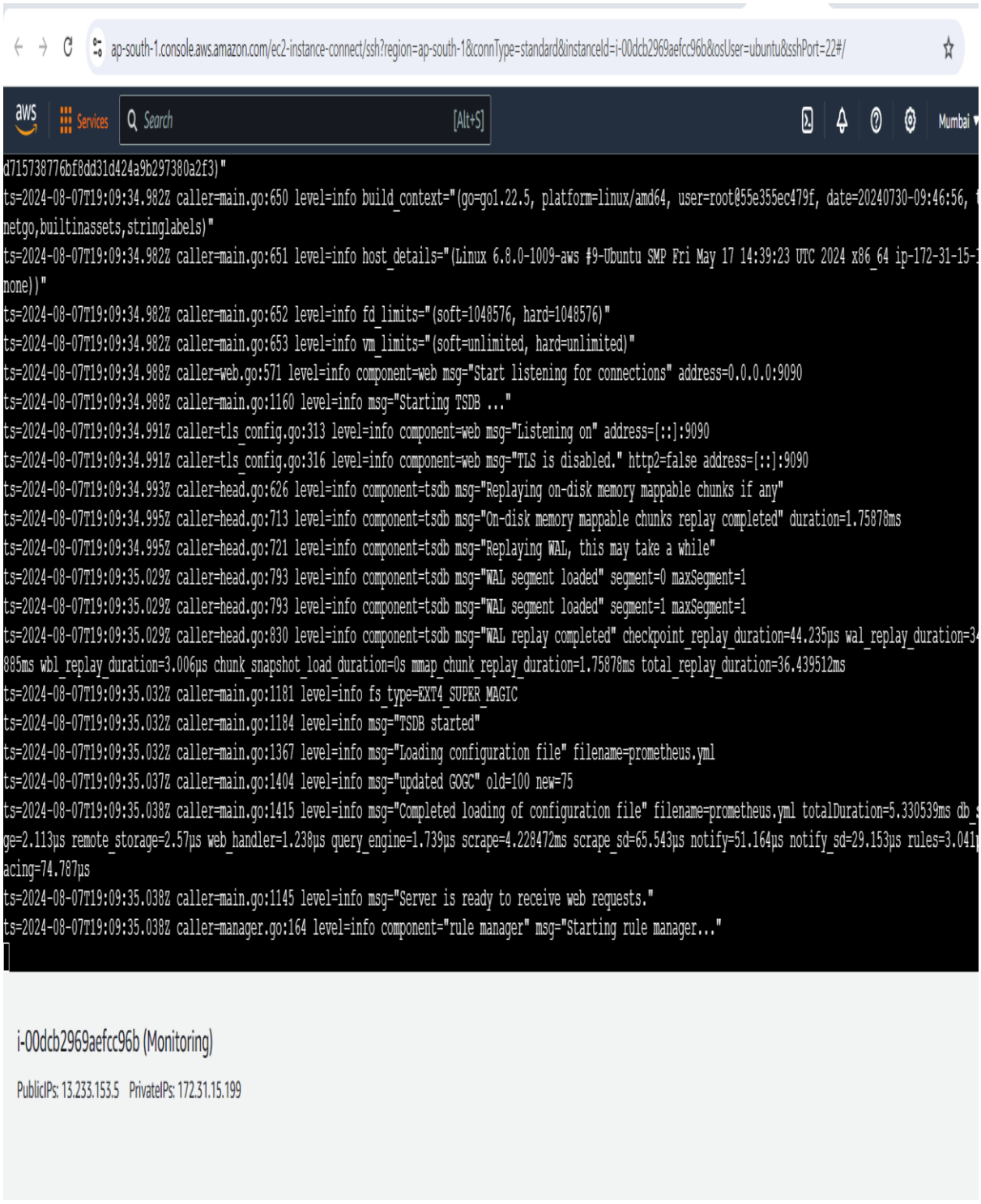
- Above ip is Host machine ip in which node exporter installed and 9100 is node selecter port



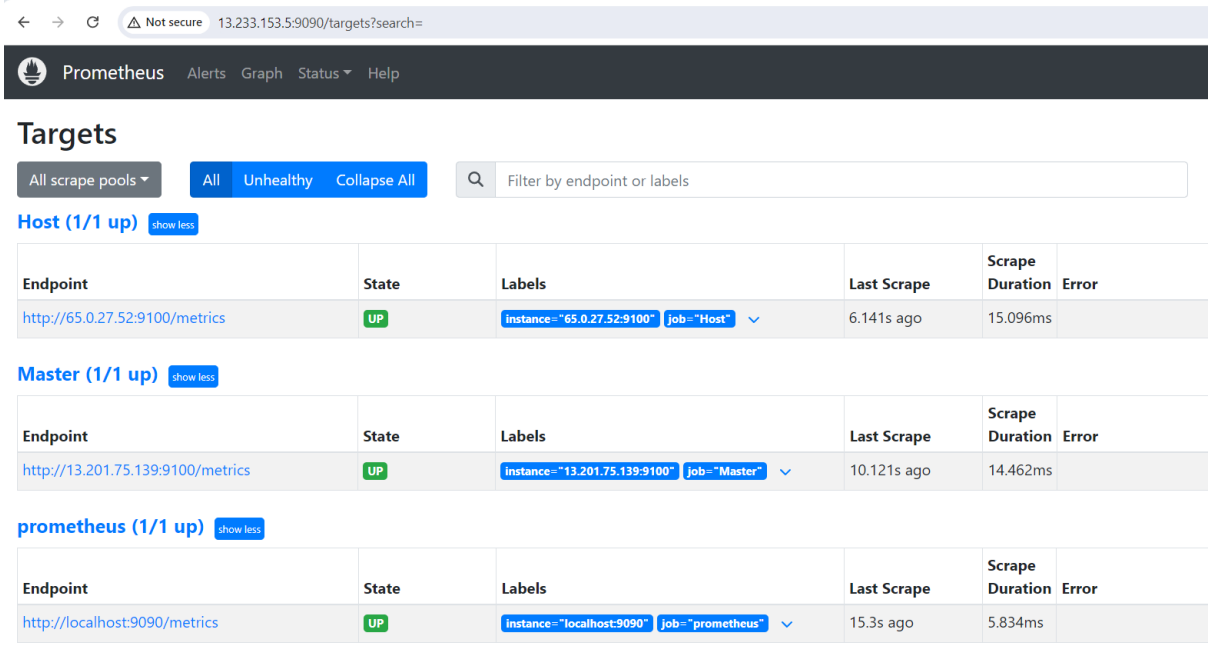
- **Save this file (press esc -----> :wq)**

Step 41: Now execute below command to run Prometheus

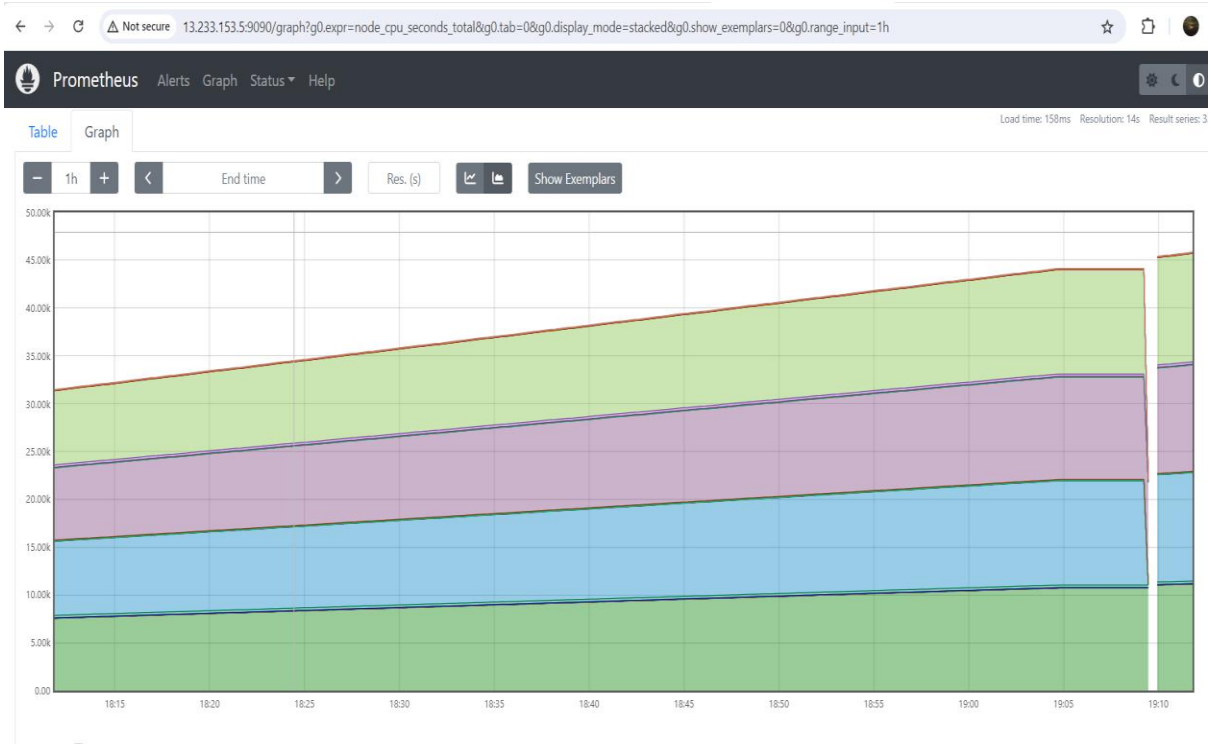
$./Prometheus



Step 42: Now go to the Prometheus page on browser and refresh the page

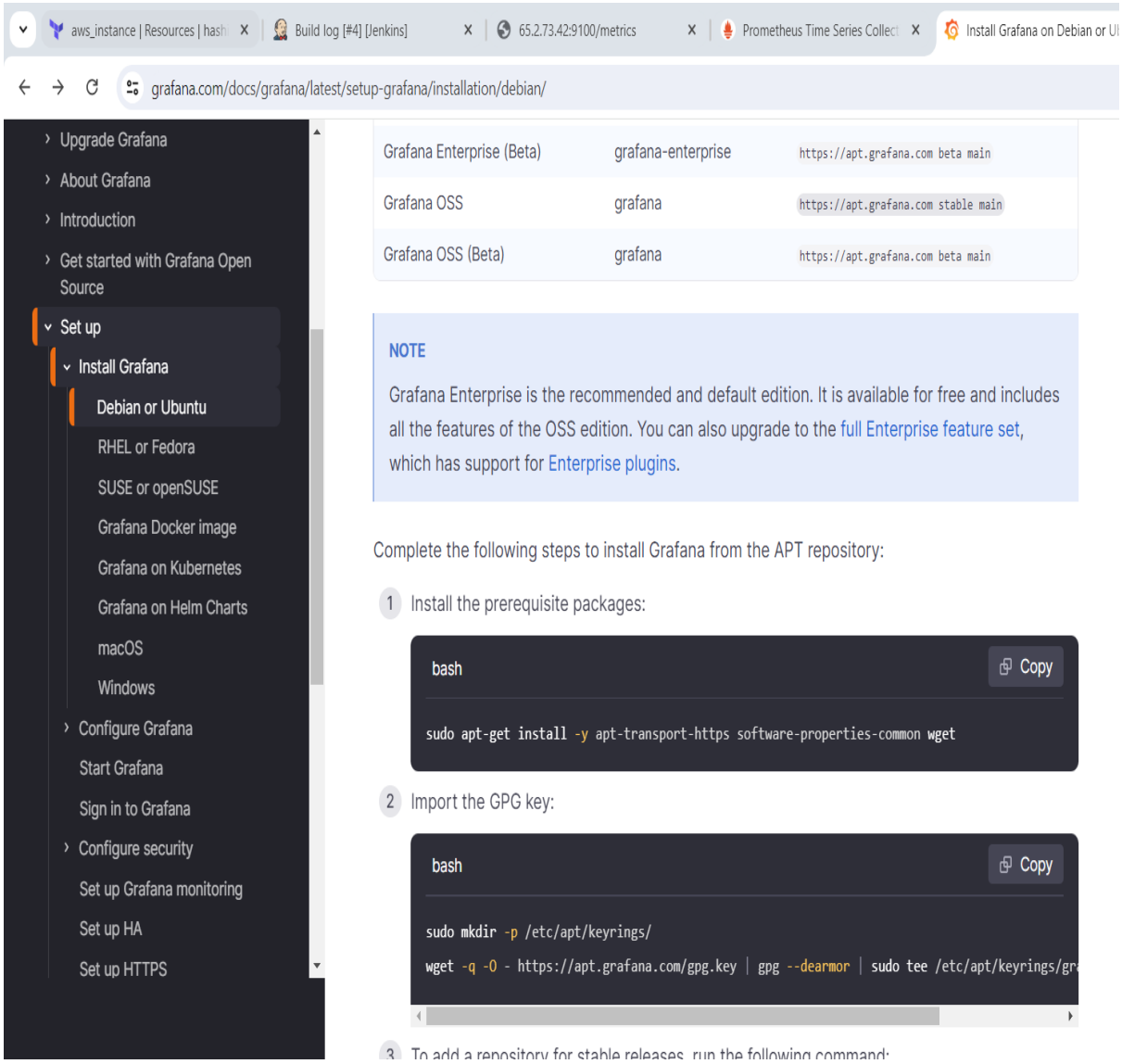


- We can also see the graph for this click on graph
- Now type node_cpu_seconds_total (for cpu utilization),then click on graph



Step 43: Now installing graphana for create dashboard

- **For install Grafana go to browser and search install grafana for ubuntu**
- **Go to official website and scroll down ,you will get the command s for install Grafana**

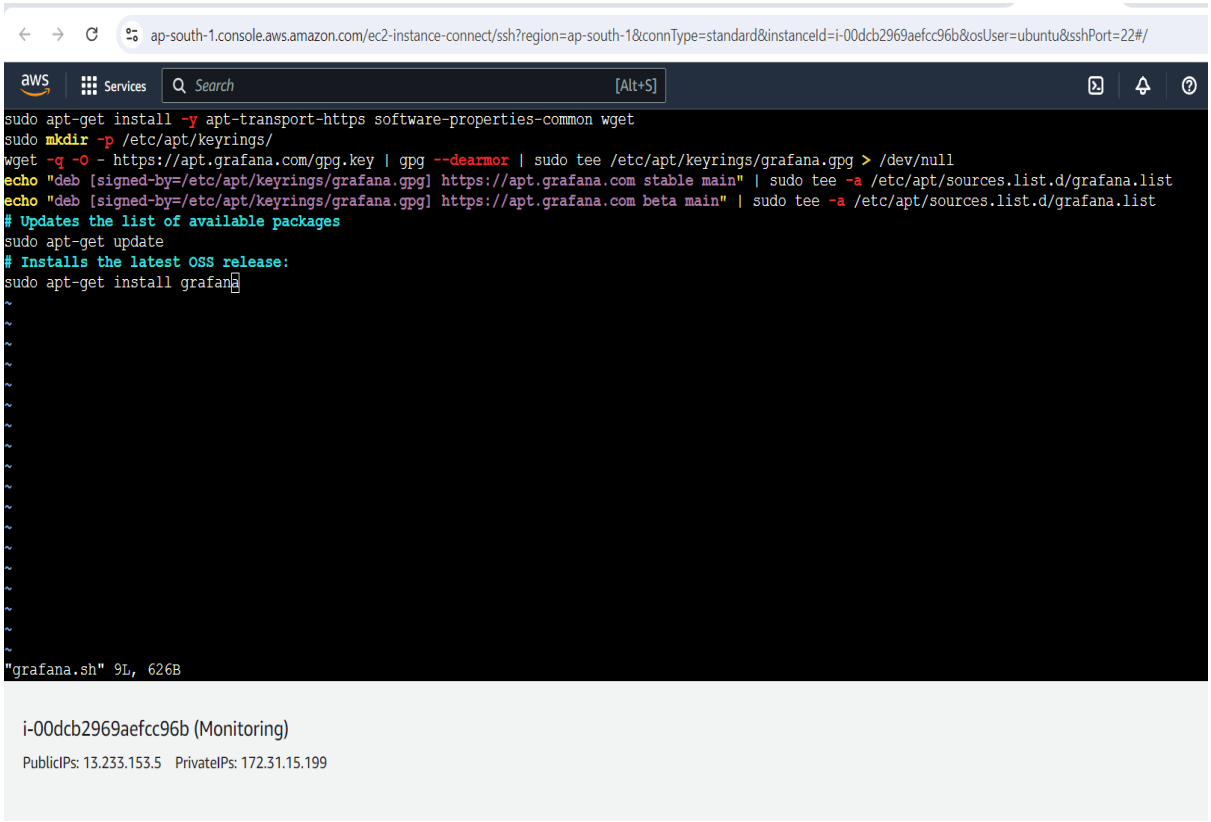


- **We will install Grafana in the same machine in which we installed Prometheus**

- **Go to the Host machine and come back one step back bye execute ($cd ..) and then create a file by execute below command**

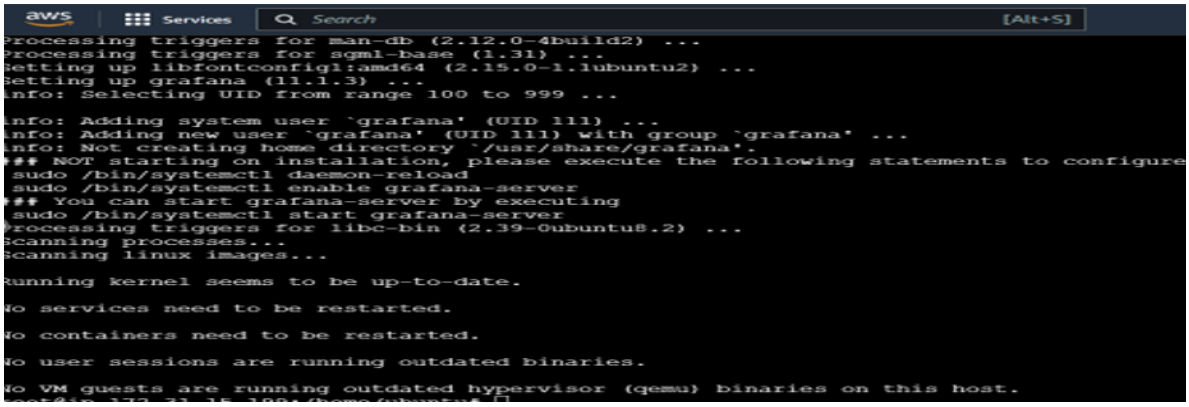
$vi grafana.sh

Now copy and paste one by one command in grafana.sh file



- **Now Execute below command to run this file**

$sh grafana.sh



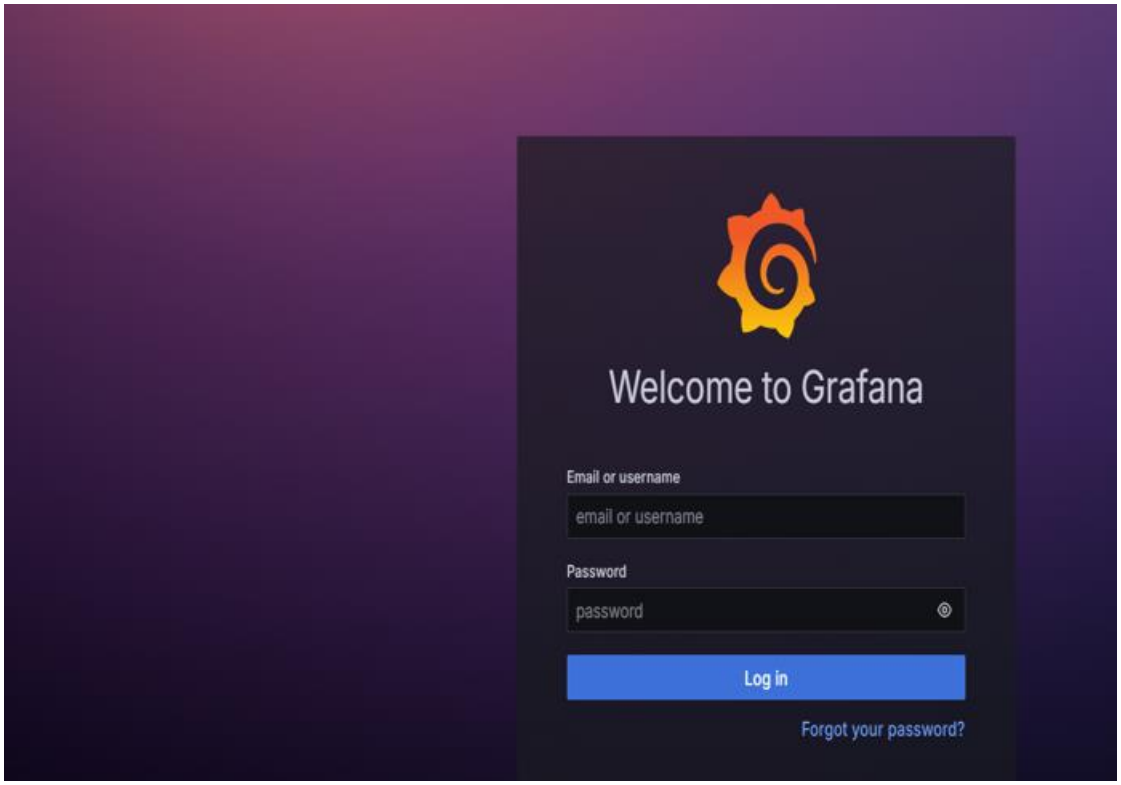
Step 43: Now execute below command in same instance

$sudo /bin/systemctl start grafana-server

```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@ip-172-31-15-199:/home/ubuntu# sudo /bin/systemctl start grafana-server
root@ip-172-31-15-199:/home/ubuntu#
```

- **Now copy the public ip of Monitoring machine instance and browse it**

13.233.153.5 :3000 (Grafana default port is 3000)

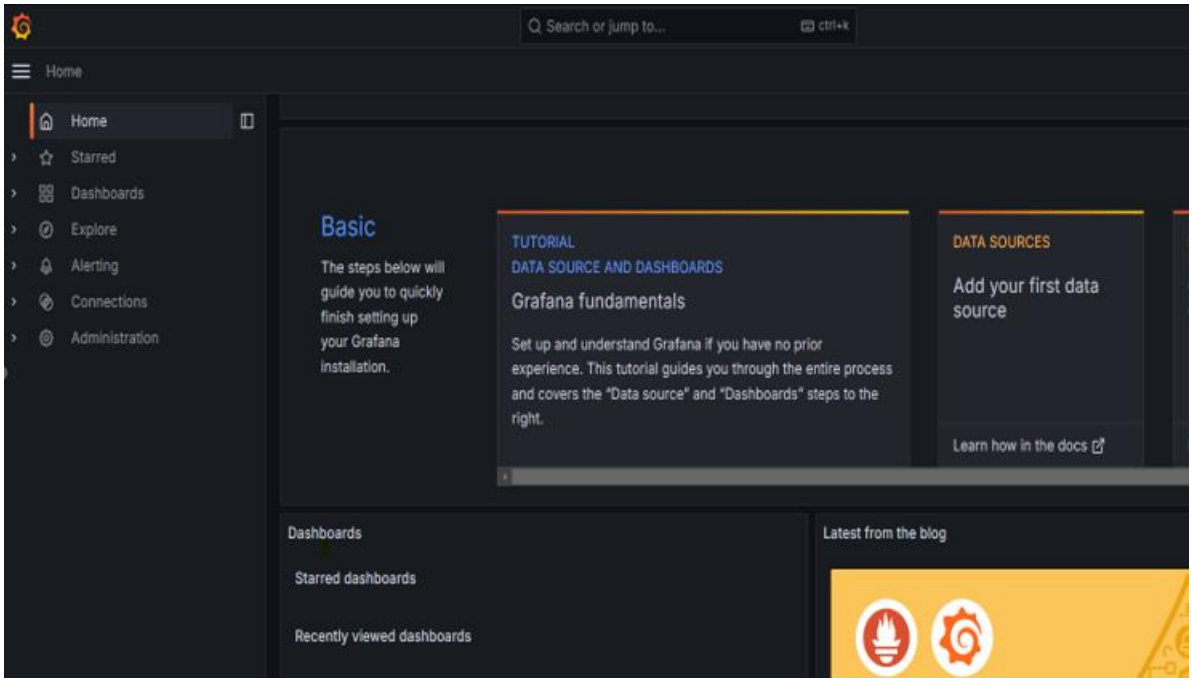


- **You can see the Grafana login page**
- **For login**

Use user name: admin

Use password : admin

- **Then click on skip**



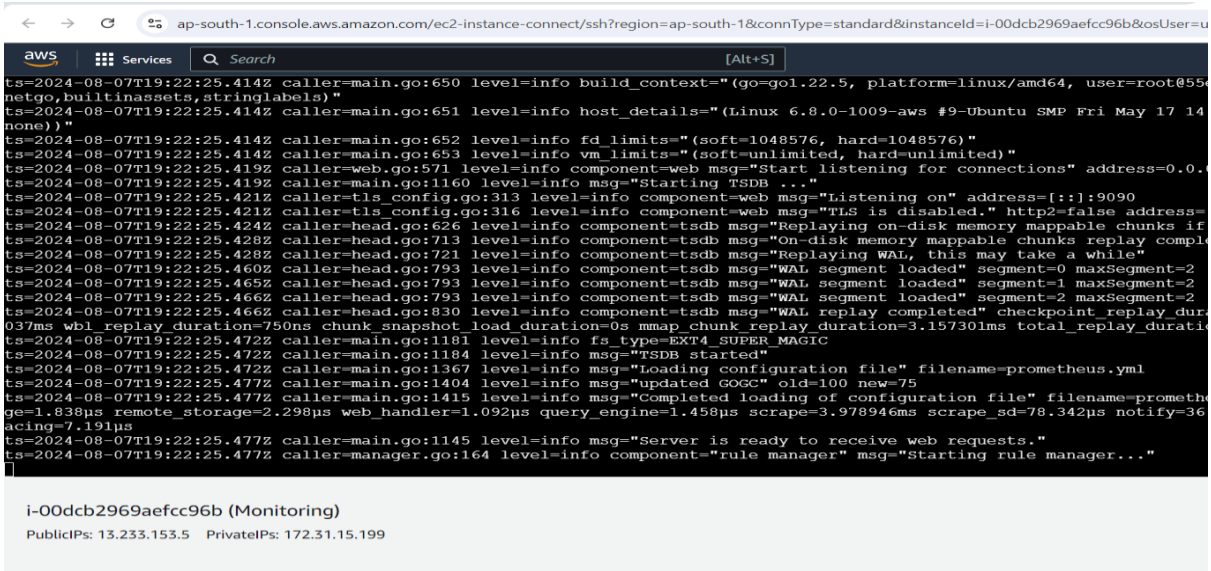
- **You can see the grafana home page**

Step 44: Now again in Monitoring Machine execute below command to run Prometheus

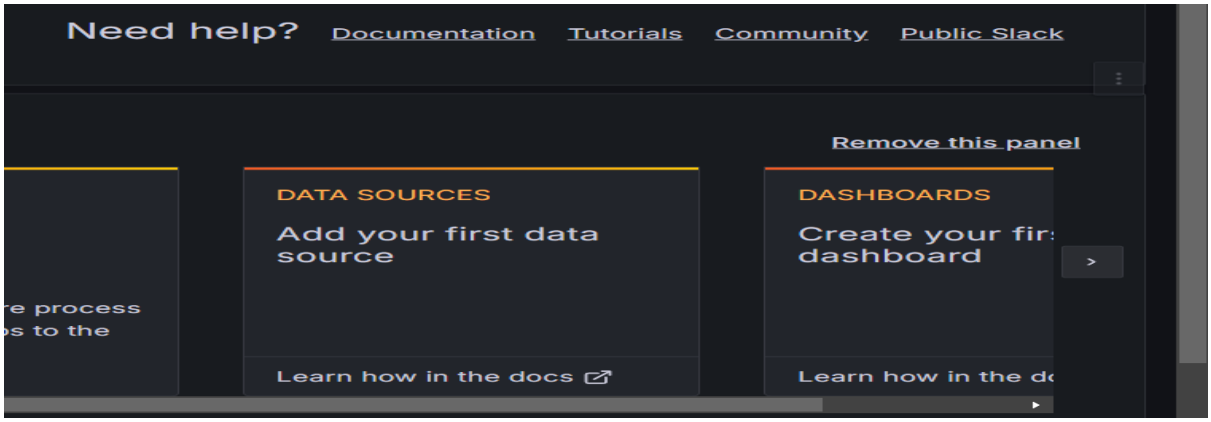
$ls

$cd Prometheus-2.54.0-rc.0.linux-amd64

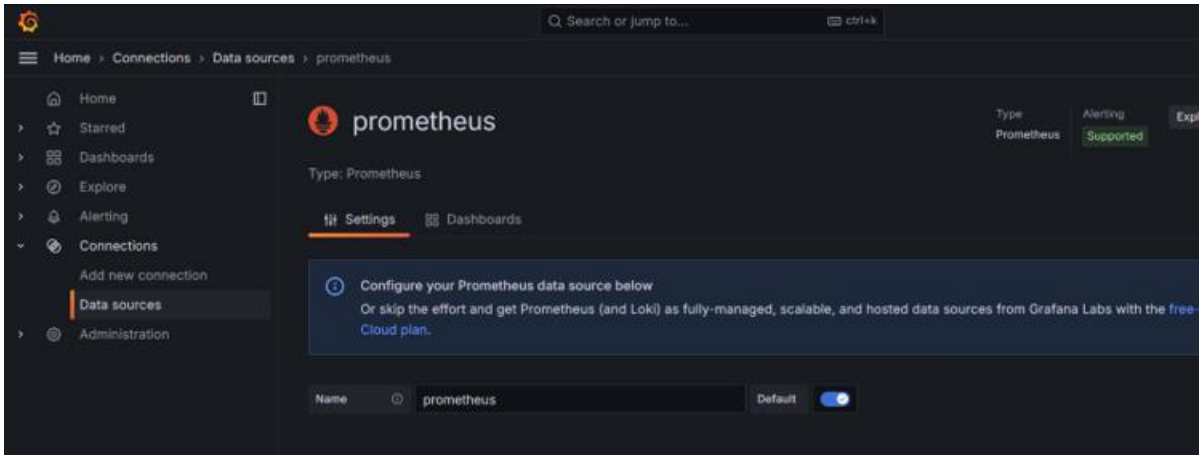
$./Prometheus



Step 45: Now click on Data source(add your first data source)

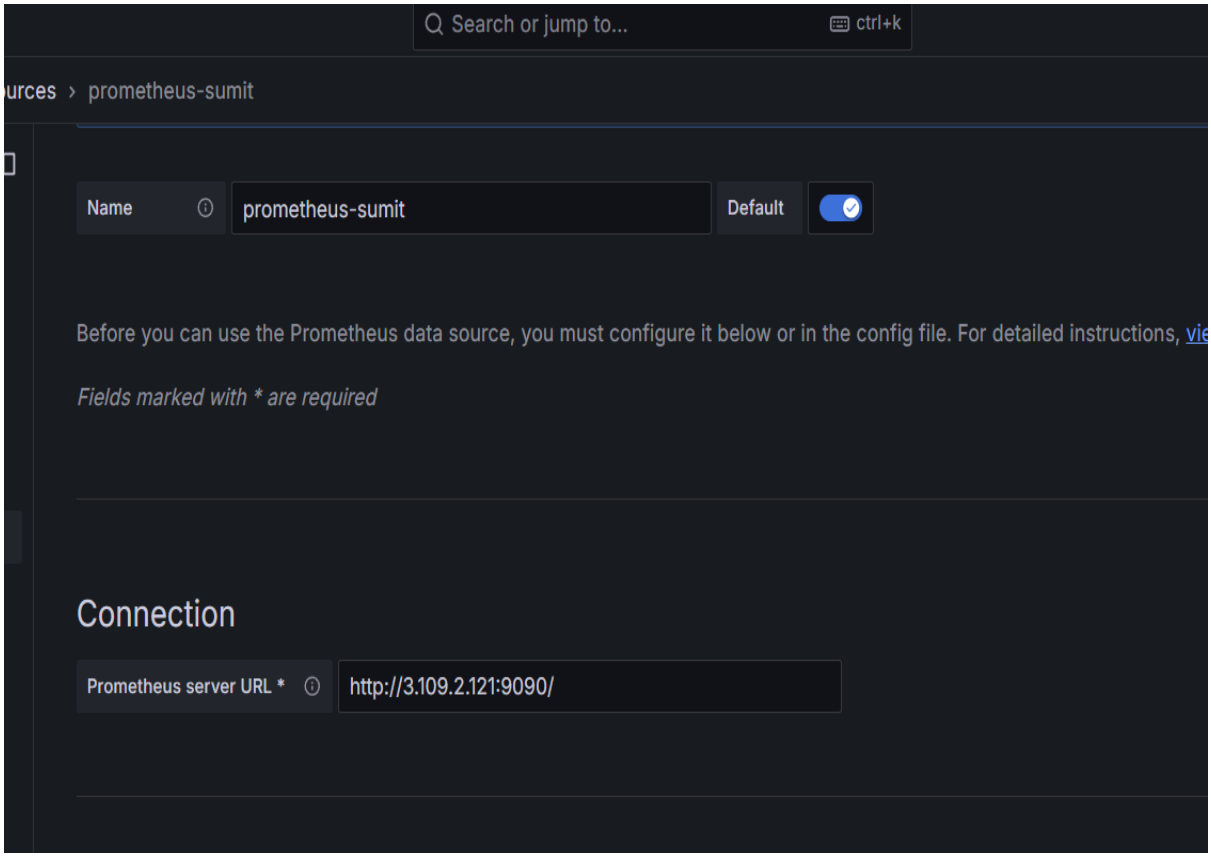


- **Then select the Prometheus as data source**

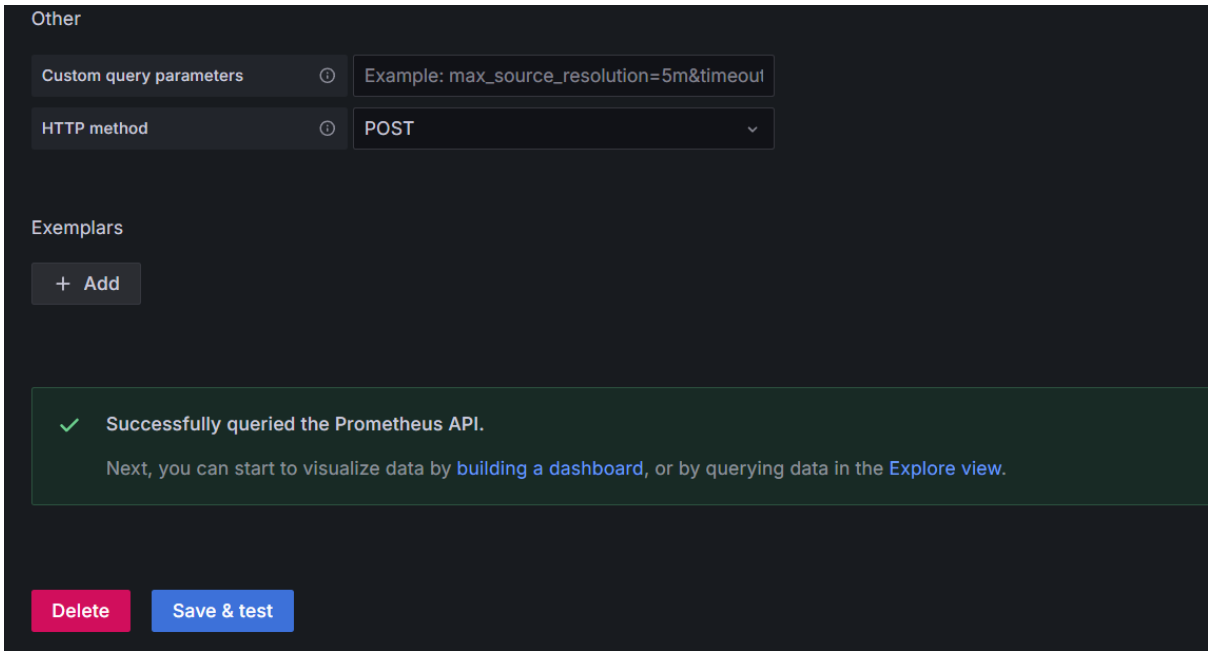


- **Name as Prometheus-sumit**
- **Now copy the link of Prometheus and paste it into connection section**

13.233.153.5 :9090/

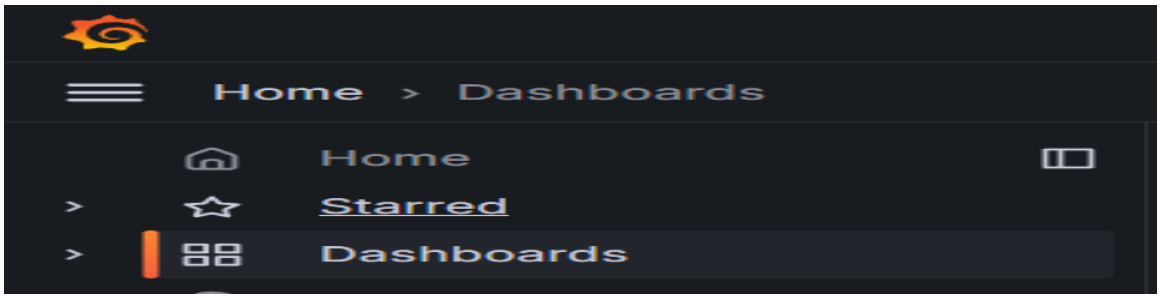


- **Scroll down and click on save and test**

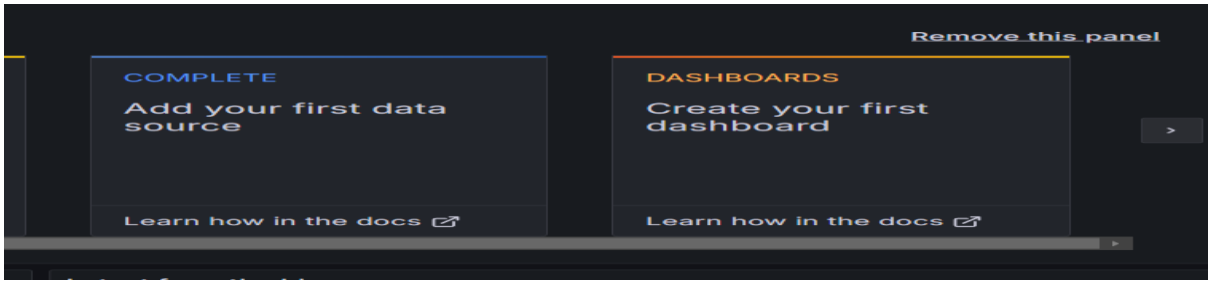


- **You can see successfully queried the Prometheus API**

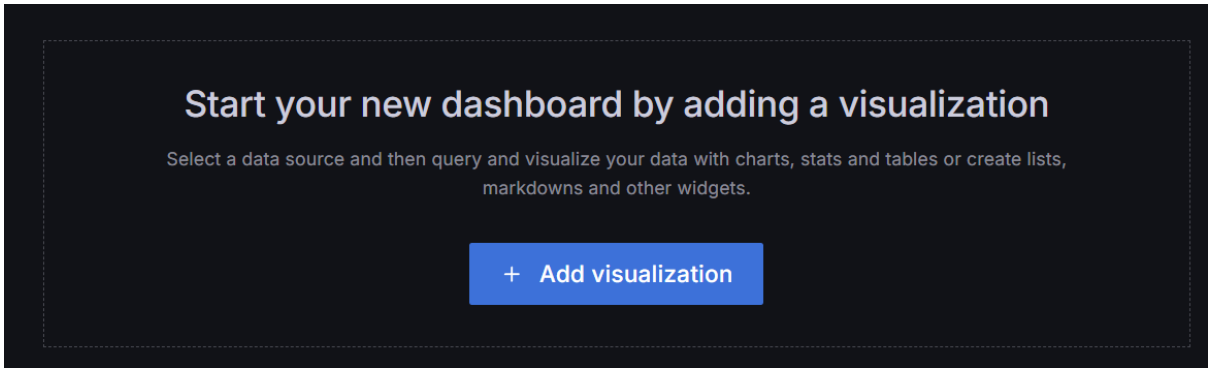
- **Now click on Dashboard**



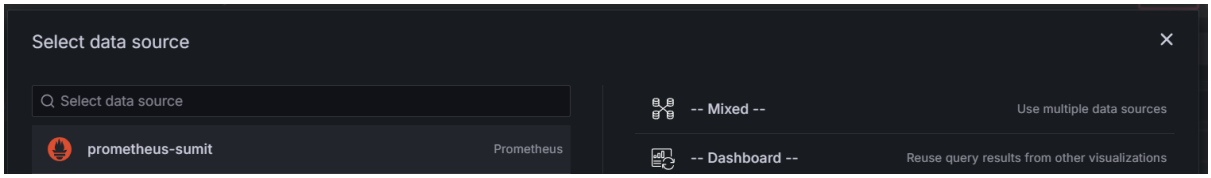
- **Now click on create your first dashboard**



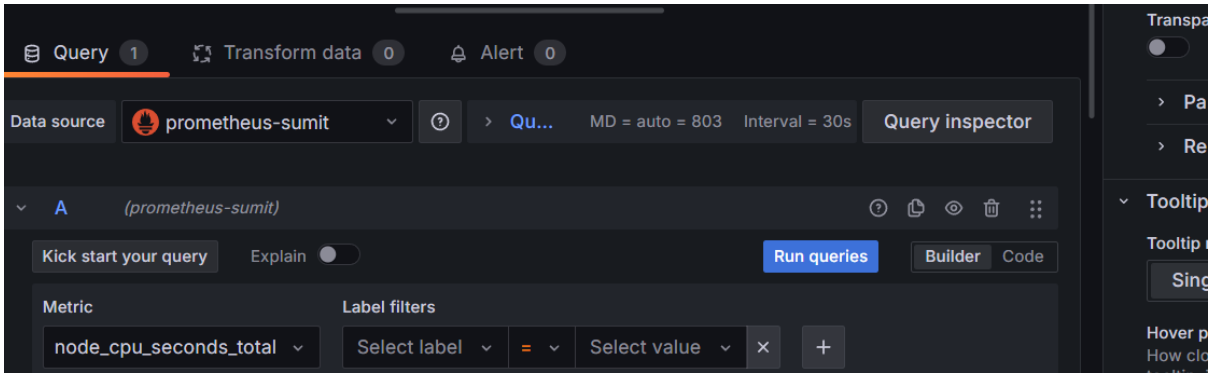
- **Then click on add visualization**



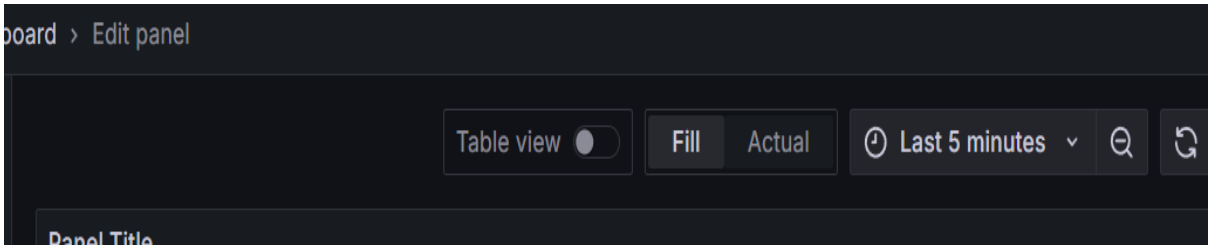
- **Select Prometheus-sumit that we have created**



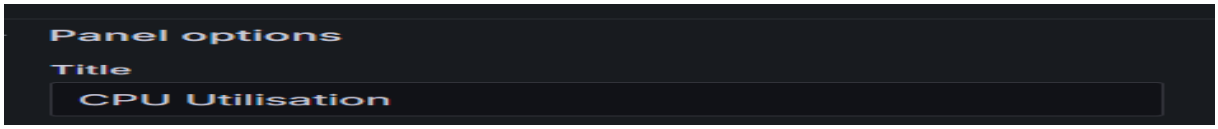
- **Select node_cpu_secods_total then click on Run queries**



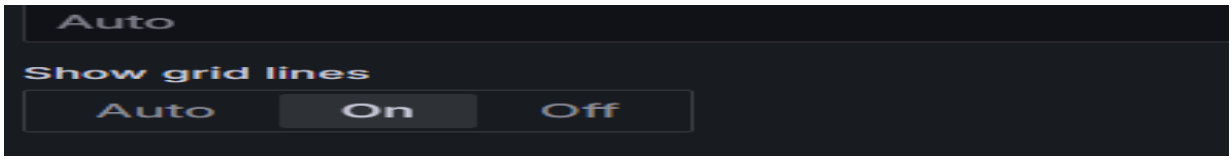
- **Select Last 5 minutes**



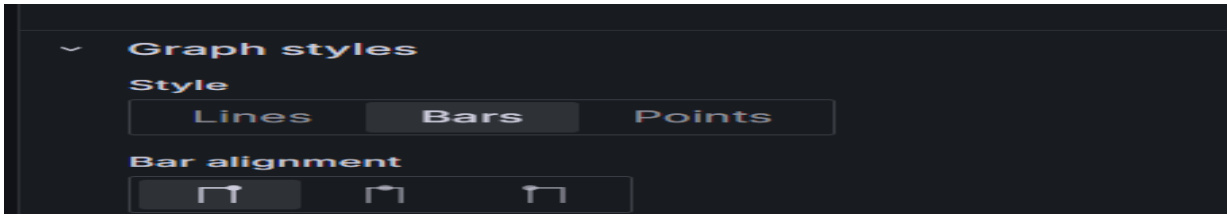
- **Given title name as CPU Utilization**

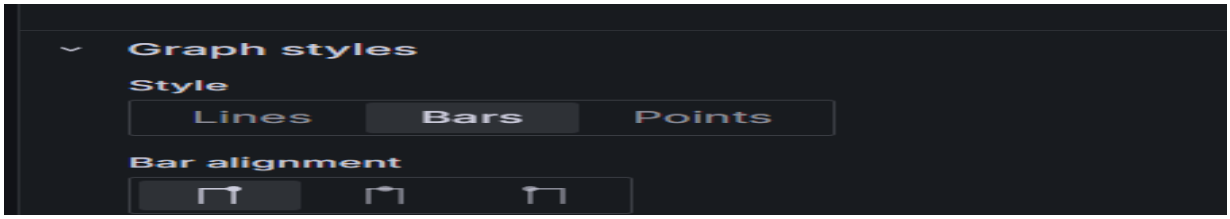


- **On the Grid lines**



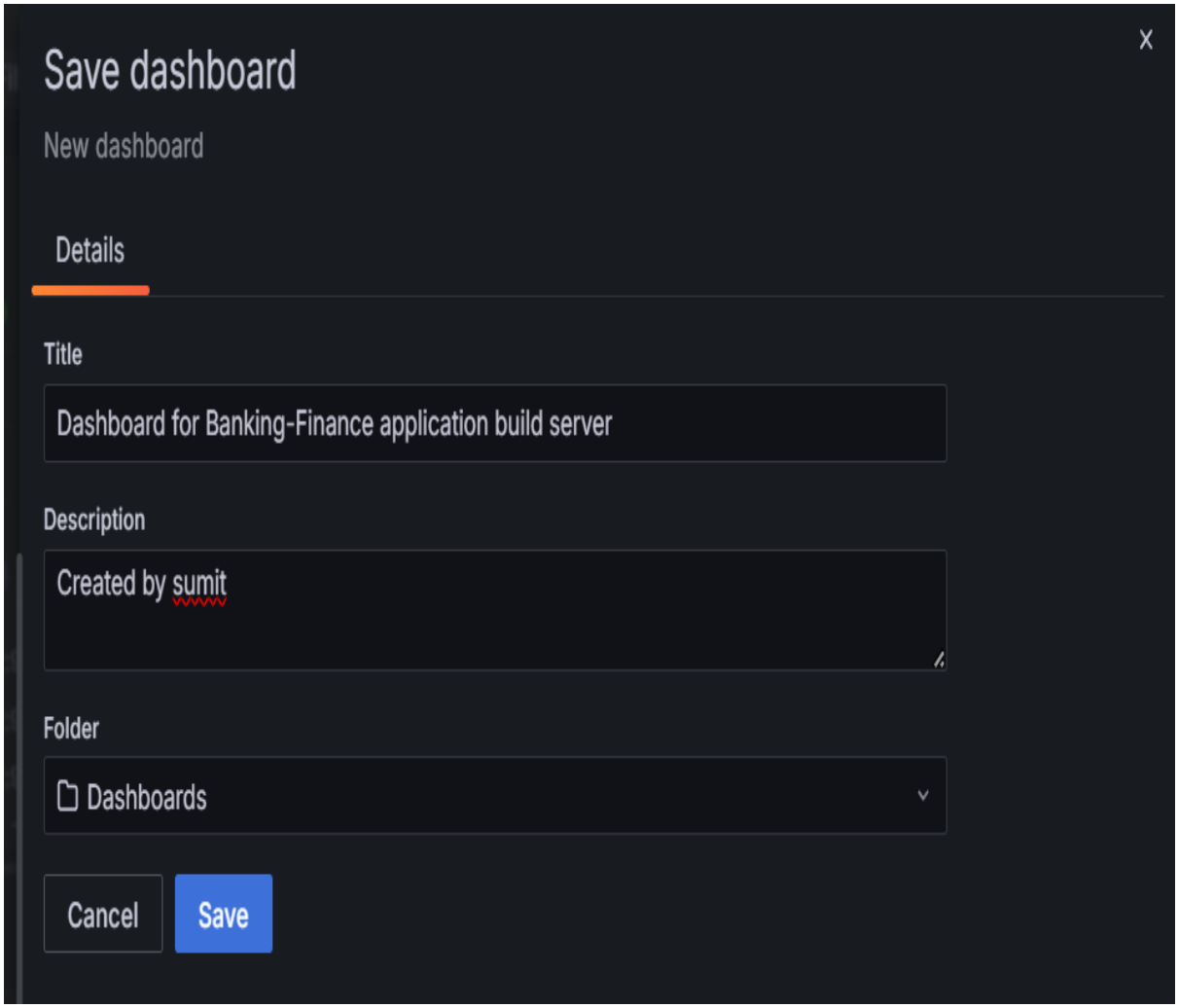
- **In style select Bar**



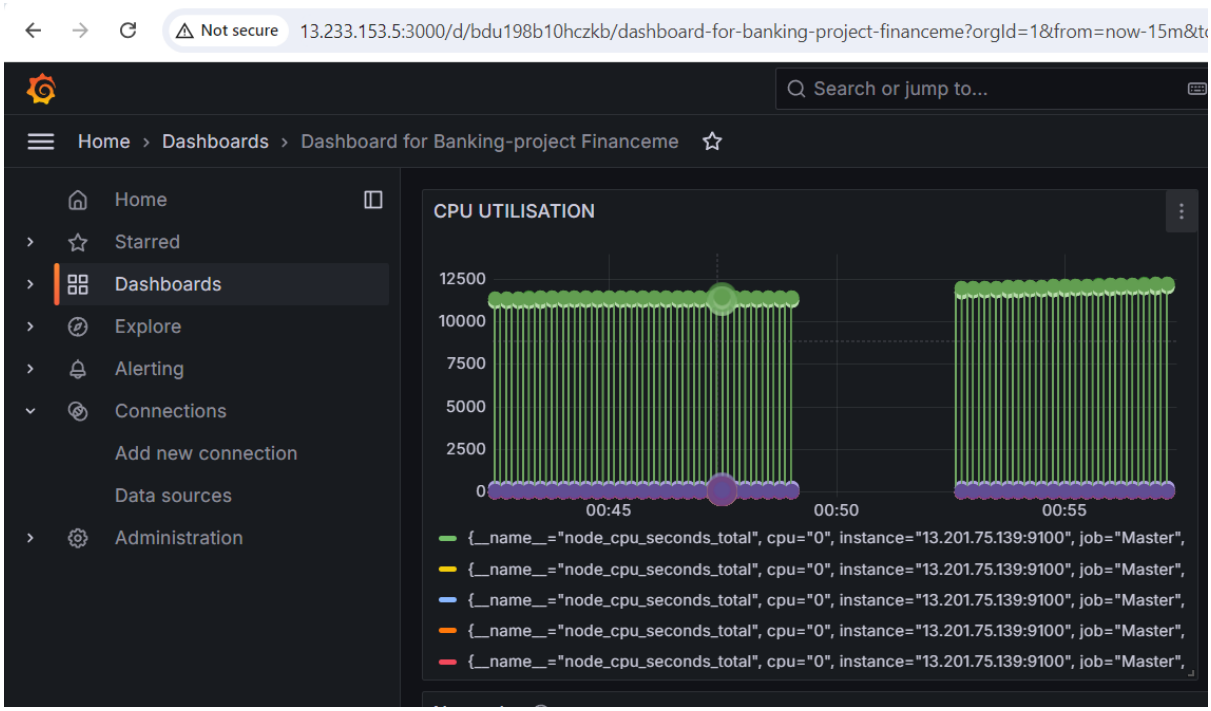


- **Show point always and Increase point size**

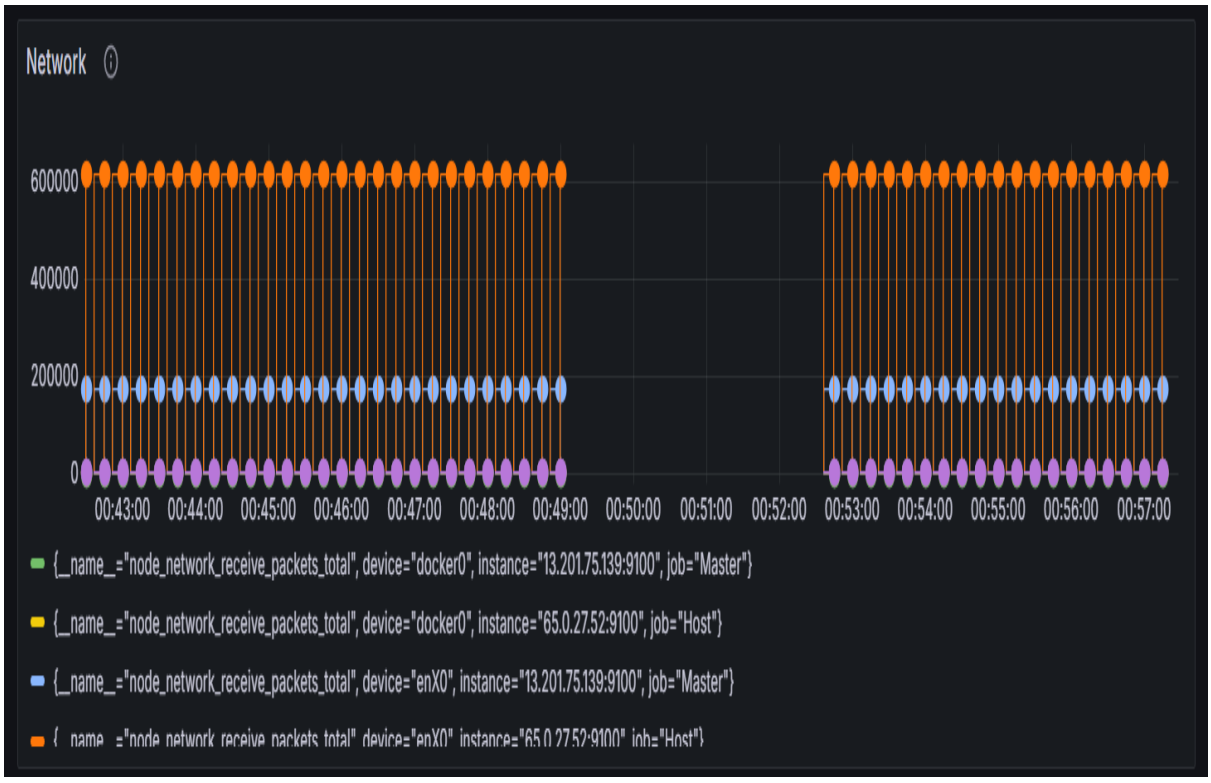
- **Click on save and Give title then click on save**



- **Now you can see the CPU utilization dashboard for Banking-Finance application build server**

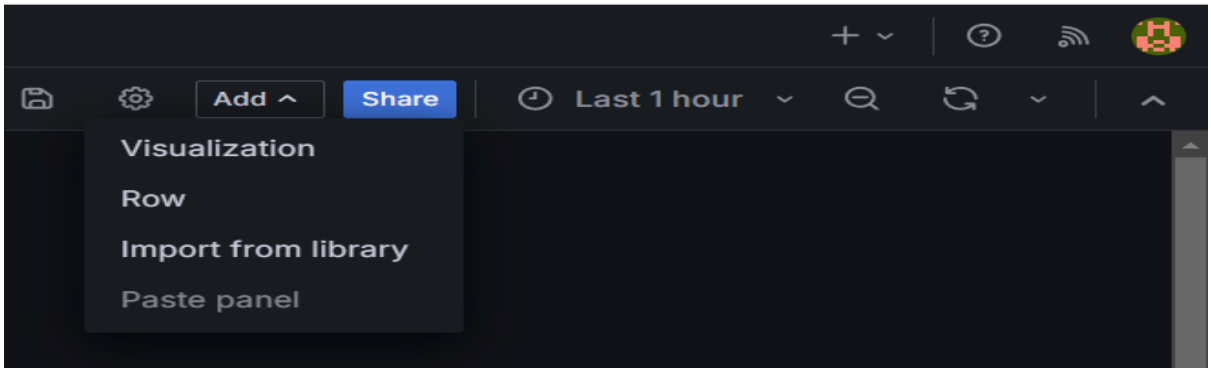


- ❖ **Now click on add to add another visualization for network**
- ❖ **click on save after create dashboard**
- ❖ **Now you can see the network**

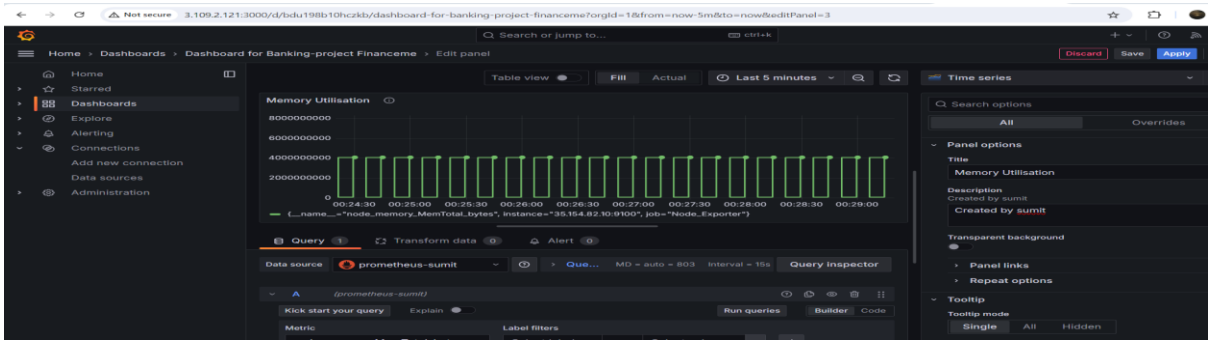


Step 46: Now adding memory utilization in dashboard

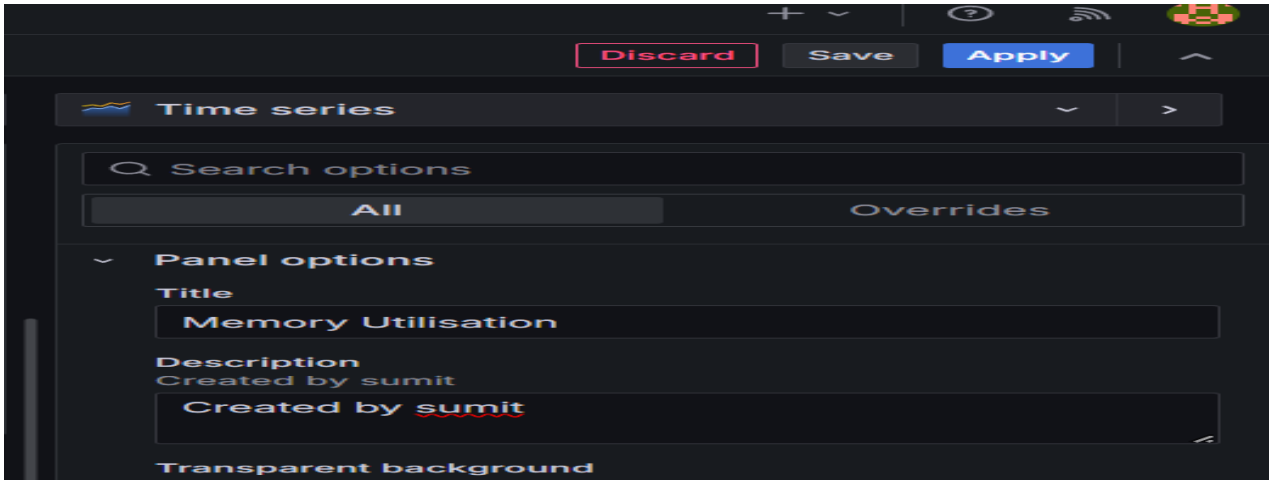
- Click on add



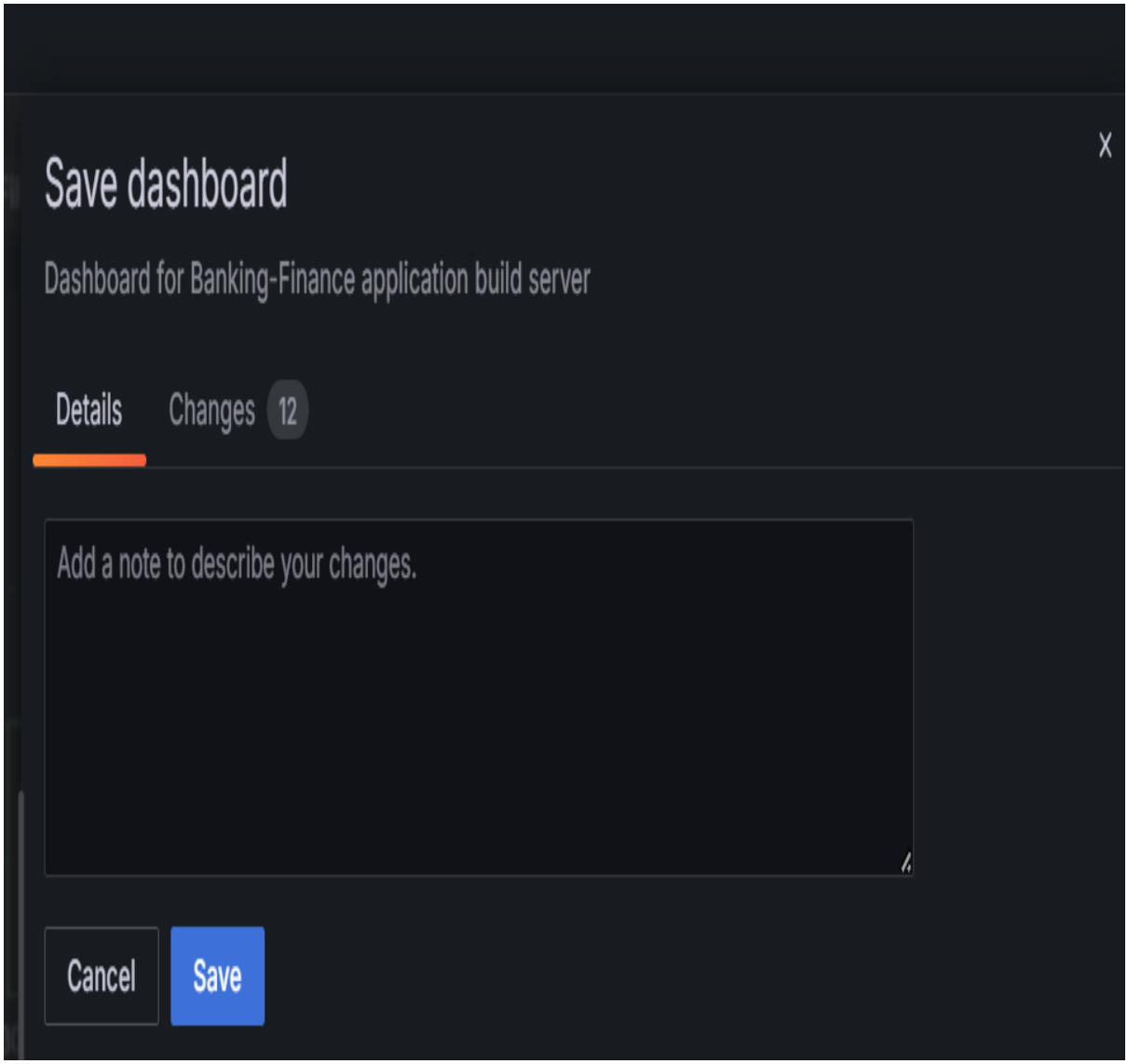
- Then select visualization
- In metrics type: node_memory_MemTotal_bytes and click on run queries ,select last 5min



- Give title as Memory utilisation and set every thing as before

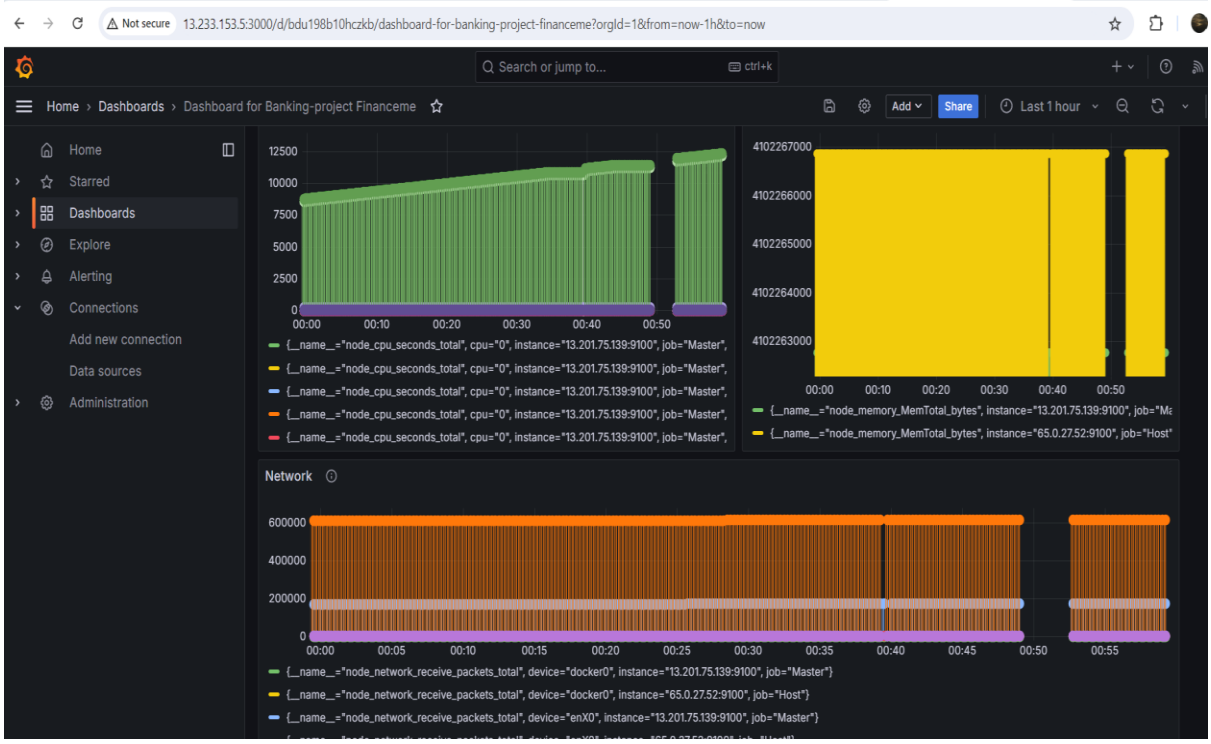


- **Click on save**

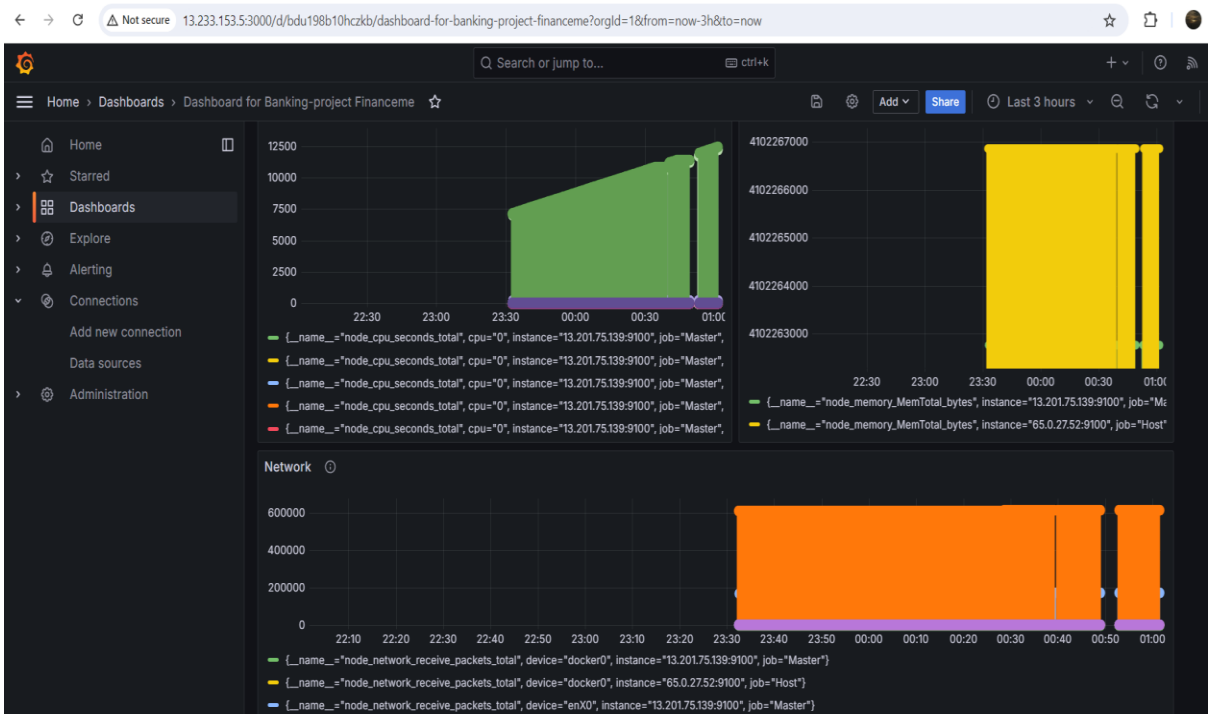


Step 47: Click on dashboard ,You can see we have created a dashboard for monitor CPU utilization ,Memory Utilization,and Network

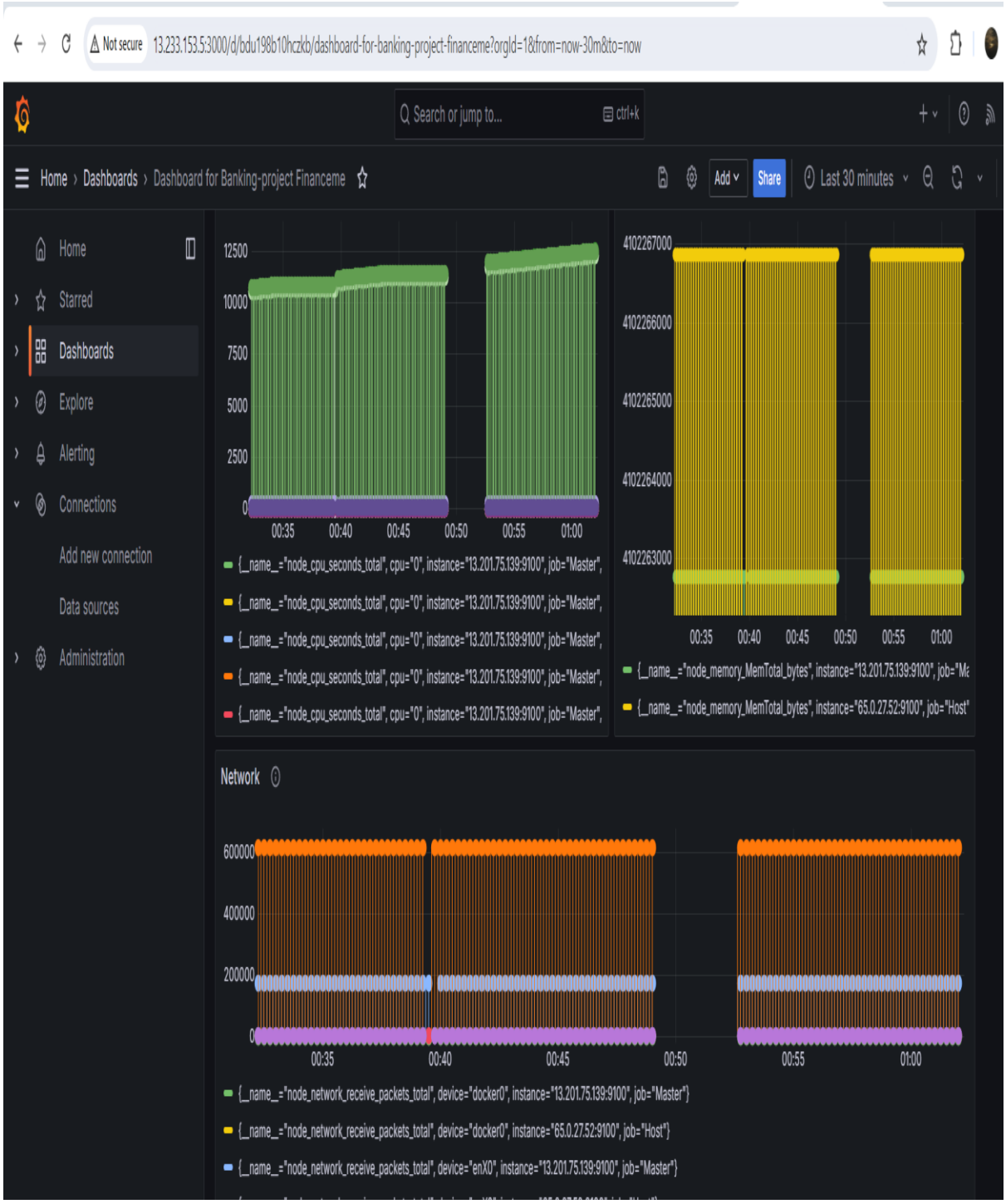
- **For last 1 hour-**



- **For last 3 hours-**

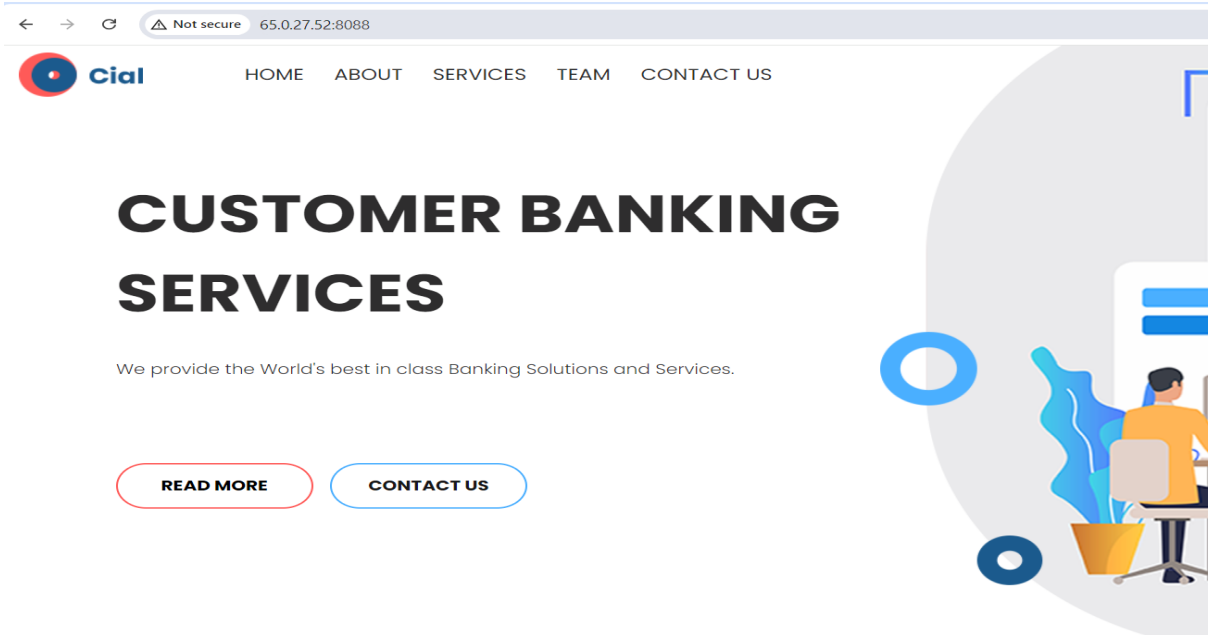


- **For last 30 minutes-**

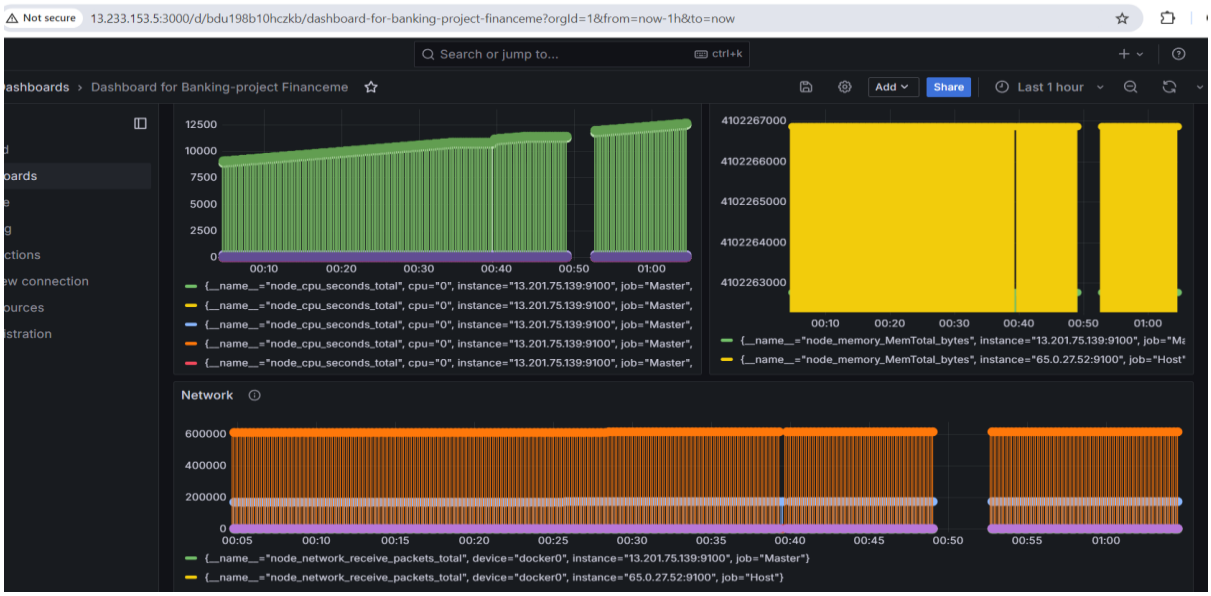


❖ **Hence we have created a Banking_project web application By using jenkins ,ansible via Jenkins,docker,Git,dockerhub etc. and we have also monitored the CPU Utilisation, Memory utilization as well as network of build server**

- **Banking-project Financeme is accessible-**



- **Dashboard for monitoring(CPU Utilisation,Network and Memory utilization)**



❖ **Hence we have created a Banking-project web application(financeme),which is accessible on internet we have also created dashboard for CPU Utilization,Network and Memory utilization of build server**